Compute (Bridgend) Ltd

CBLe Text Editor User Manual

8 Merthyr Mawr Road, Bridgend, Wales UK CF31 3NH

Tel: +44 (1656) 65 2222
Fax: +44 (1656) 65 2227

CBL Web Site - http://www.cbl.com

This document may be downloaded from http://www.cbl.com/cblidoc.html

CBL Ref: c:\nbj\htmldoc\td\CBLe_Text_Editor_Manual_FrontPage.html - 2008/05/29 15:45:02 -

Contents

Contents

Contents

Introduction to the CBL Text Editor

The CBL text editor (CBLe) may be used to perform text edit on the following:

- z/OS PDS/PDSE members.
- z/OS Sequential Data Sets.
- z/OS HFS Files.
- CMS files.
- VSE LIBR members.
- VSAM KSDS, ESDS, RRDS, VRDS, AIX & PATH.

The CBLe edit environment is a hybrid of the IBM ISPF Editor and CMS XEDIT, supporting command syntax and data display indicative of both edit environments.

The user interface is modelled on PC style Multiple Document Interface (MDI) standards (although limited by restrictions imposed by 3270 display).

Since its inception, the CBLe text editor has been expanded beyond the scope of simple text edit. In its current form, CBLe also supports the CBLVCAT execution window and all other list and dialog windows that have traditionally been the domain of the CBLi main window only. See CBLi documentation for complete description of these window types and their usage.

For users who have a SELCOPY licence, CBLe also supports the Structured Data Environment (SDE), a more advanced type of edit for data sets whose records have a pre-determined structure as defined by a COBOL or PL/1 copy book. This feature enables the user to edit data within the strict confines of the defining structure with records being displayed as a number of individual fields. This subject is described in detail in the CBLi Structured Data Environment (SDE) documentation.

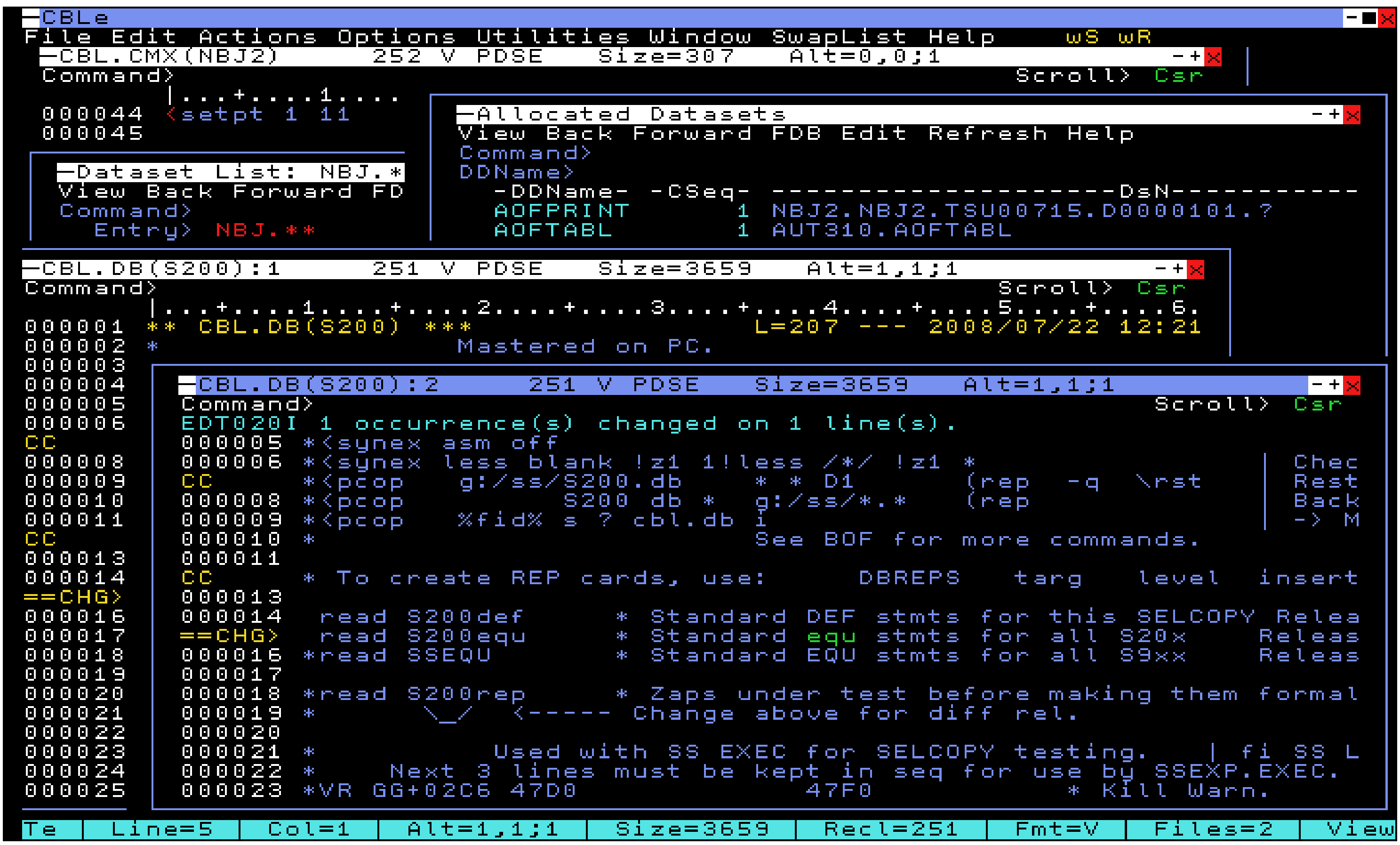

Figure 1. CBLe Window.

CBLe Main Window

The CBLe main window (also called the CBLe MDI frame window) has a standard CBLi format title bar with system menu, minimise, maximise, restore (if applicable) and close buttons.

The client area of the window contains a menu bar on the first line of the display and an information bar for the currently edited file on the last line of the display. Both lines are always in view.

The remainder of the client area contains the document (child) windows, one for each CBLe text edited file view, list window, SDE edit view, etc.

If the CBLIINI file variable **Edit.Instance** is set to **Multiple** or the CBLe SET option INSTANCE MULTIPLE is in effect, then any number of CBLe main windows may be opened within the CBLi session.

By default, the MDI frame windows of all MDI applications supported by CBLi (i.e. CBLe and SELCOPY Interactive) are opened in a maximised state if the 3270 terminal display width is 80 bytes or less.

Figure 2. CBLe Main Window.

CBLe Main Window Menu Bar

The CBLe main window menu bar contains the following items:

File

New	Edits a new document window.
Open ...	Opens the Open dialog window allowing the user to browse for the file to be edited.
Close	Close the current document window.
Structured edit ...	Open the SDE Edit dialog.
Save	Save the file in the current document window with the current fileid.
Save As ...	Save the file in the current document window and prompt for fileid.
Execute SELCOPY ...	Open the window.
Execute CBLVCAT ...	Open the Execute CBLVCAT (VCI) window.
DB2 Dynamic SQL	Open the DB2 Dynamic SQL window.
Allocate NonVSAM ...	Open the Allocate NonVSAM dialog.
Define KSDS ...	Open the Define KSDS dialog.
Define ESDS ...	Open the Define ESDS dialog.
Define RRDS ...	Open the Define RRDS dialog.
Define LDS ...	Open the Define LDS dialog.
Execute IDCAMS ...	Open the IDCAMS Command window.
Create Library ALIAS	Execute the ALIAS -DLG CBLi CLI Command to open the Create ALIAS window for Library members.
Exit	Exit the current CBLe main window.

Edit

Undo	Undo change levels in the current document window.
Redo	Redo change levels that have been undone in the current document window.
Cut	Remove currently marked text and place in the CBLi clipboard.
Cut append	Remove currently marked text and append it to existing data in the CBLi clipboard.
Copy	Copy currently marked text to the CBLi clipboard.
Copy append	Append currently marked text to the CBLi clipboard.
Paste	Paste text from the CBLi clipboard to the last cursor position.
Select All	Mark all text in the file area of the current document window.
Reset Block	Reset marked text in the file area of the current document window.
Find ...	Open the Find dialog to search the data in the current document window for a specified text string.
Change ...	Open the Change dialog to search the data in the current document window for a specified text string and replace occurrences with a new text string.
System Edit ...	Open the default system editor (XEDIT, ISPF Edit) to edit the current file.

Actions

Sort ...	Open the Sort dialog to sort data in the current document window.
Fill ...	Open the Fill dialog to fill a marked block.
Uppercase	Upper case all text in the currently marked block.
Lowercase	Lower case all text in the currently marked block.

Options

Query SET Options	Display settings of all available QUERY command options with the exception of QUERY MACRO.
Edit SET Options	Edit a temporary CMX command file containing a SET command for all available SET options.

Utilities

List	Open a sub-menu of List window items:

DASD Volumes ...	Open the DASD Volumes list window.
Cataloged files ...	Open the Cataloged files list window.
Dataset details ...	Open the Dataset details list window.
VTOC files ...	Open the VTOC files list window.
VTOC extents ...	Open the VTOC extents list window.
Allocated files ...	Open the Allocated files list window.
Library members ...	Open the Library members list window.
Enqueues ...	Open the Enqueues list window.
Job Enqueues ...	Open the Job Enqueues list window.
HFS Dir Path ...	Open the HFS Path list window.

System	Open a sub-menu of System window items:

Operating system ...	Open the Operating System window.
CBLi storage stats ...	Open the Stoarge statistics Block window.
CBLi module list ...	Open the Module List window.
CBLi SVC ...	Open the CBLVCAT SVC information window.
CBLNAME ...	Open the CBLNAME storage display window.

File search ...	Open the File Search window.
File search/update ...	Open the FSU - File Search/Update window.
Calendar ...	Open the Calendar window.
Calculator ...	Open the Calculator window.
SDSF	Under TSO or ISPF, call the SDSF JES2 spool access program.
ISPF Dataset Utilities	Under ISPF, call the ISPF dataset utilities menu panel.

Window

New Window	Open a document window containing a new view of the file in the current document window.
All Windows	Open the Window List window. Select new focus window.

Cascade	Cascade the document windows.
Tile Horizontally	Tile the document windows horizontally.
Tile Vertically	Tile the document windows vertically.
Arrange..	Open a popup menu to arrange current edit views horizontally/vertically. (Not activated.)
Arrange Minimised	Arrange the minimised document windows.
title1 ... titleN	Lists all document windows within the the CBLe main window. Select new focus document window.

SwapList

Displayed only if ISPF is the 3270 screen manager. Execute the ISPF SWAPLIST command.

Help

Contents	Open the help documents page.
CBLi Environment	Open the CBLi environment help contents page.
CBLe Text Edit	Open the CBLe Text Edit help contents page.
SDE Edit	Open the Structured Data Environment help contents page.
SELCOPY Manual	Open the SELCOPY user manual.
CBLVCAT Manual	Open the CBLVCAT user manual.
About CBLi ...	Display CBLi Release and Build level information.

CBLe Main Window Status Bar

The CBLe main window Status bar contains the following items:

Te/Se	Current (or last visited) edit window is a **Text Edit** window view or **Structured Edit** window view.
Line=*n*	Line number of current line in current document window. "?" is displayed if the line number is unknown. (e.g. SDE KSDS edit by KEY)
Col=*n*	Column number of current column in current document window.
Alt=*n1,n2;x*	Alterations made since last save or autosave, last save and number of undo levels. An "*" (asterisk) indicates that Redo levels may be applied.
Size=*n*	Number of records in the file within the current document window. **Size>***n* is displayed if the SDE edit technique does not automatically read all records from the file and the data is displayed for edit before end-of-file is reached.
Recl=*n*	Lrecl of the file within the current document window.
Fmt=*X*	Record Format (RECFM) of the file within the current document window.
Files=*n*	Number of files being edited in this CBLe main window.
Views=*n*	Total number of views of all files edited in this CBLe main window (equal to number of document windows)
yyyy/mm/dd HH:MM:SS	Current date and time in columns 91 to 109. Since its position is beyond column 80, this is only visible on dynamic 3270 terminals with a greater number of columns than that provided by a standard 3270 terminal.

CBLe Text Edit Document Window

The CBLe main (frame) window is an MDI parent window which supports a number of different types of document window, e.g. SDE Edit view, Execute CBLVCAT window, Data Set List window, etc. The format and capabilities of these types of window are discussed in relevant sections of the CBLi Reference documentation and Structured Data Environment (SDE) documentation.

It should be noted that the SELCOPY Interactive feature of CBLi is also an MDI frame window which supports all the same document windows supported by the CBLe frame window, including CBLe text edit documentation windows.

This CBLe Text Edit documentation focuses exclusively on properties and command syntax applicable to CBLe text edit document windows.

A CBLe text edit document window (edit view) has a standard CBLi format title bar with system menu, minimise, maximise, restore (if applicable) and close buttons. The title bar also contains the fileid, LRECL, RECFM, DSORG, file size (number of records) and alteration count for the file being edited. Furthermore, the literal **(Read Only)** is displayed in the title bar if the user does not have read/write authority for the file. Therefore, where the title bar of a non-maximised edit view is visible, the characteristics of the file being edited may be determined without having to make the edit view the focus window.

The text edit document window also contains a command prompt and, potentially, a vertical and horizontal scroll bar all of which are not considered part of the client area.

The client area of a document window contains a horizontal scale in the first line of the display and, optionally, a prefix area displaying line numbers. Before and after the editable text, the client area includes a "Top of File" and "End of File" marker.

The remainder of the client area is the file display area and contains the editable text.

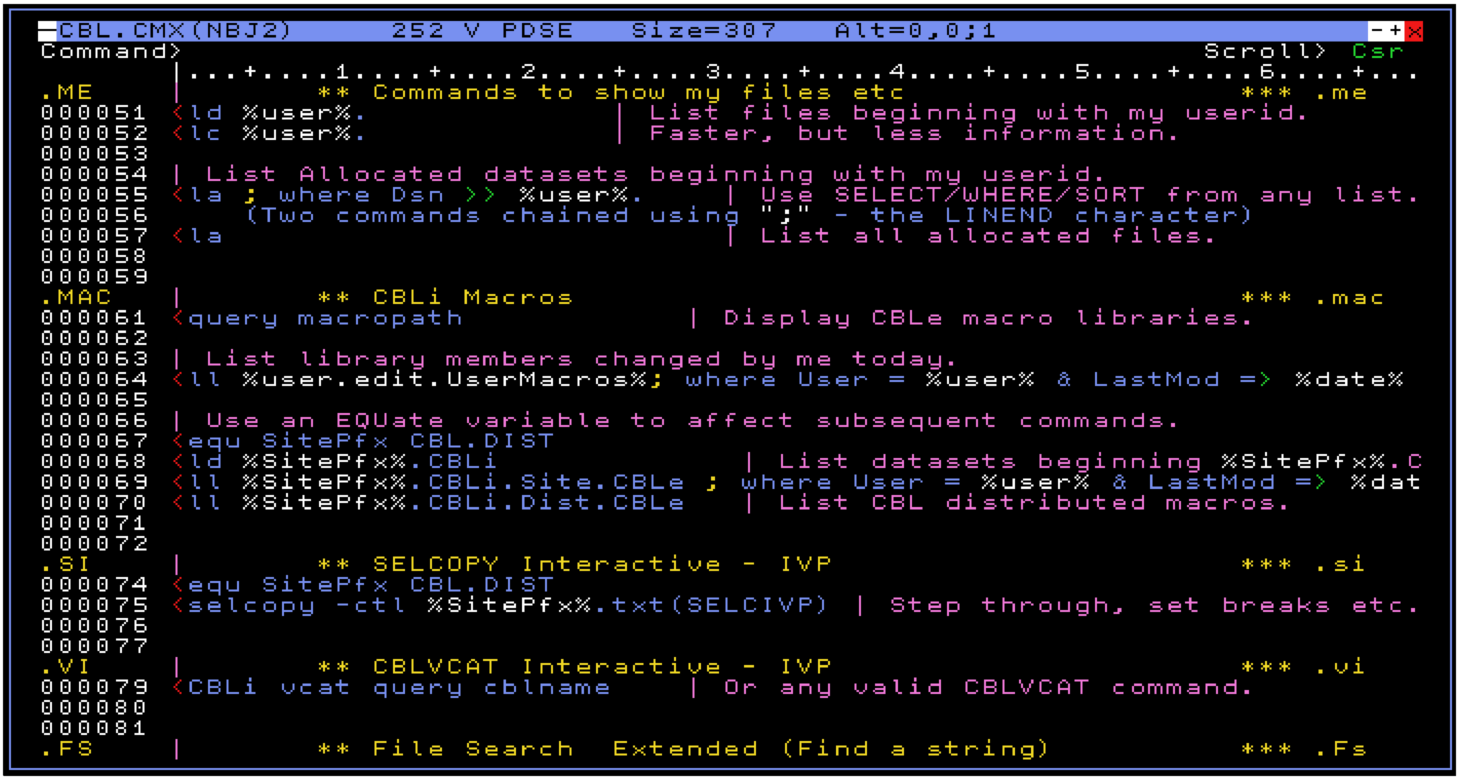

Figure 3. CBLe Text Edit Document Window.

CBLe Current Window

The current CBLe window is the CBLe text edit document view on which focus was last placed. This may be the focus window or, if the focus window is not a CBLe text edit document view, the last CBLe text edit view visited.

The concept of a current CBLe window view is important when executing CBLe CLI commands from an SDE window display via the TEDIT SDE CLI command or selecting CBLe main menu bar items.

In both cases, the current CBLe window will be the target of the operation.

CBLe SDE Document Window

In addition to CBLe text edit document windows, the CBLe frame window also supports Structured Data Environment SDE window views as one of its MDI child (document) windows.

SDE is supported in MVS environments only where a SELCOPY licence is active.

An SDE edit/browse window view may be opened to edit/display data set records that have an associated structure generated from a COBOL or PL1 copybook, using one of the following methods:

1. Select "Structured edit" to open the SDE edit dialog window from the "File" item of the CBLe main menu bar.
2. From a CBLe command prompt, execute the CBLe SDATA CLI command to invoke an SDE EDIT or BROWSE CLI command.
3. Execute Prefix command "SB" - SDE Browse or "SE" - SDE Edit from any file List or Execute CBLVCAT window.

To distinguish an SDE edit document view from a CBLe text edit document view, the edit type flag, displayed on the left of the CBLe status bar, reflects the type of edit of the current (last in focus) edit view. This flag is either "Te" for CBLe Text Edit or "Se" for SDE Structured Edit.

Concepts and functionality of the Structured Data Environment are documented at length in the SDE Edit help documentation.

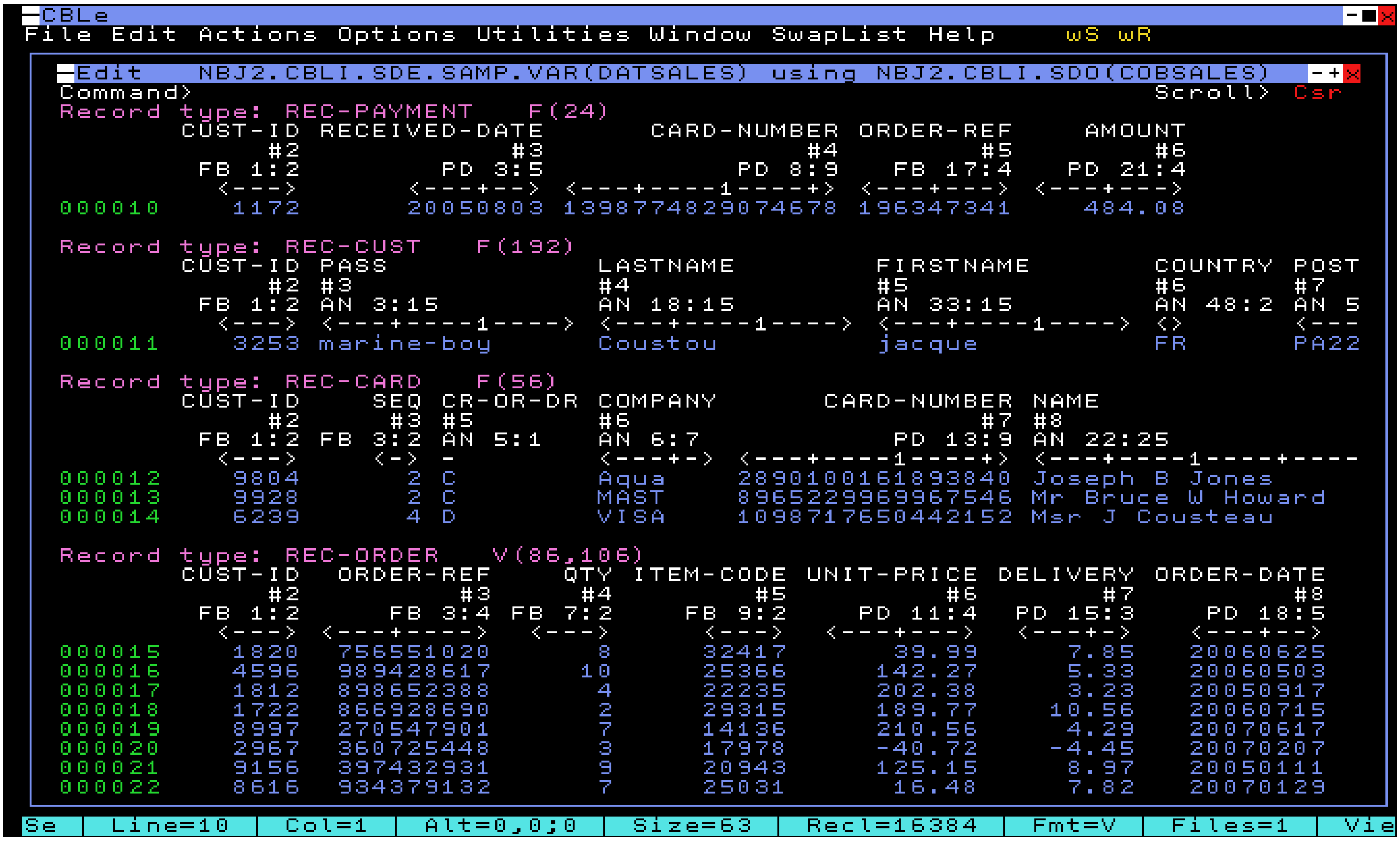

Figure 4. CBLe SDE Document Window.

CBLe List, DB2 SQL & CBLVCAT Document Windows

In addition to CBLe text edit and SDE edit document windows, the CBLe frame window also supports CBLi List windows, DB2 Dynamic SQL windows and CBLVCAT Interactive Windows as one of its MDI child (document) windows.

When MDILIST is set ON (the default), any List or Execute CBLVCAT window started from within CBLe is opened as an MDI child of CBLe. This may be done as follows:

1. Select "List" and the required list type from the "Utilities" item of the CBLe main menu bar.
2. Select "DB2 Dynamic SQL" from the "File" item of the CBLe main menu bar.
3. Select "Execute CBLVCAT" from the "File" item of the CBLe main menu bar.
4. Execute a CBLi list type CLI command from a command prompt within the CBLe environment. (e.g. LD, LC, LVOL, LA, SQL, VCAT.)
5. Execute a CBLi list type Prefix command from an existing List or Execute CBLVCAT window in the CBLe environment. (e.g. "M" - member list, "L" - List Data set, "VC" - Execute CBLVCAT.)

When MDILIST is set OFF, the List window is opened as a child of the CBLi main window.

The format and functionality of a List, DB2 SQL or Execute CBLVCAT window opened as a CBLe MDI child window is no different to that when it is opened as a child of the CBLi main window. By default, the CBLi command MDINEXT is assigned to PF9 in all CBLe child windows. This allows the user to pass focus from the current CBLe child window to the next opened CBLe child window which may be a CBLe text edit, SDE structured edit, DB2 Dynamic SQL, Execute CBLVCAT or a list type window.

List, DB2 SQL and Execute CBLVCAT windows do not support the same concepts as CBLe text edit and SDE structured edit document windows (e.g. text update, focus line, etc.). Therefore, CBLe and SDE CLI commands and macros may not be successfully issued from List and Execute CBLVCAT windows.

Similarly, all CBLe main menu bar items are specific to CBLe text edit windows (e.g. "Fill", "Uppercase", "Lowercase") and, if selected, will apply to the current CBLe window. Each List, DB2 SQL and Execute CBLVCAT window has its own menu bar items applicable to that individual window, displayed immediately below the document window's title bar.

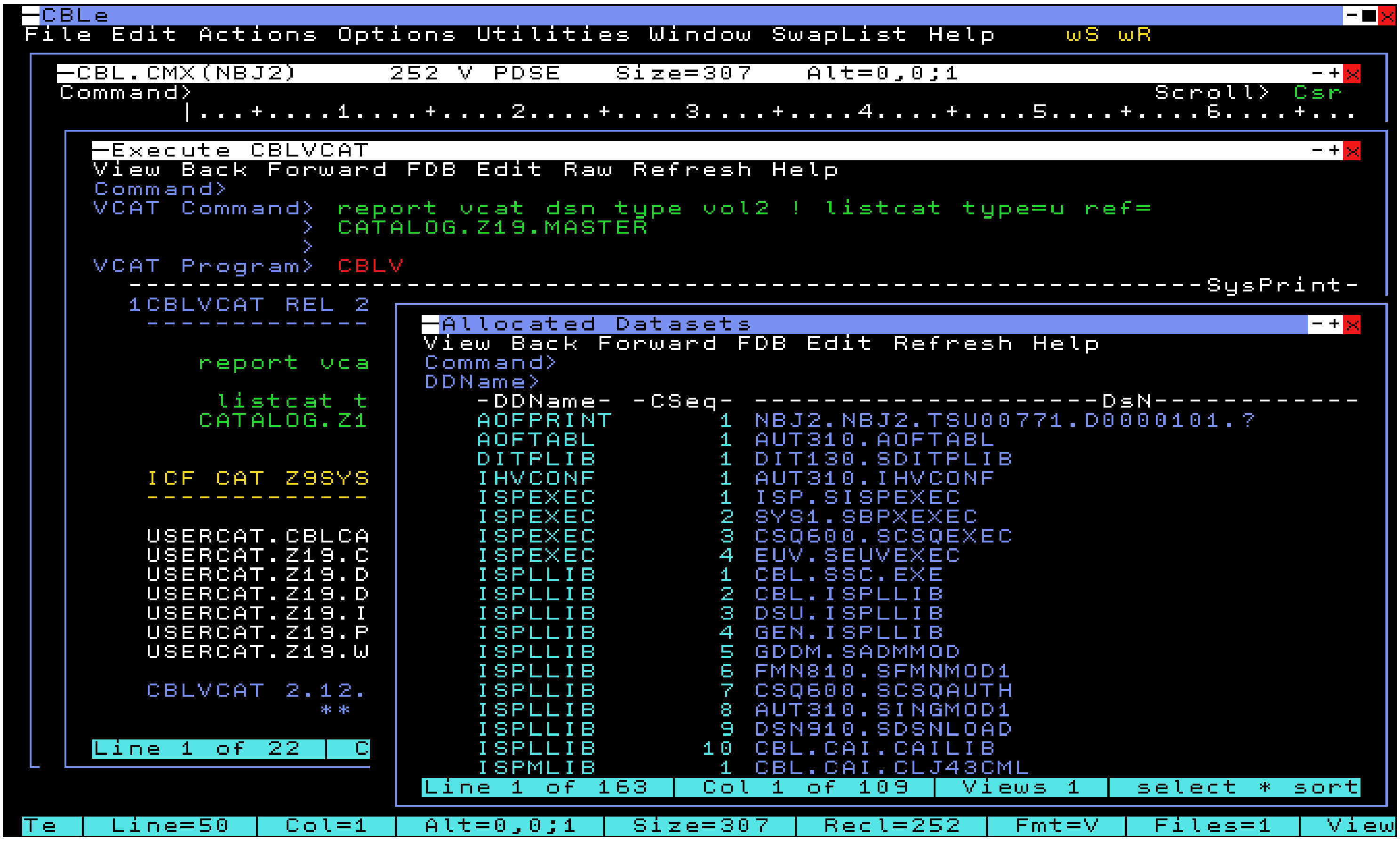

Figure 5. CBLe List & CBLVCAT Document Window.

CBLe Open Dialog Window

The CBLe text edit Open dialog window may be opened via the following:

- Select **Open** from the **File** menu item in the CBLe Main Menu Bar.
- Enter the CBLe command OPEN at the command line of any document window.

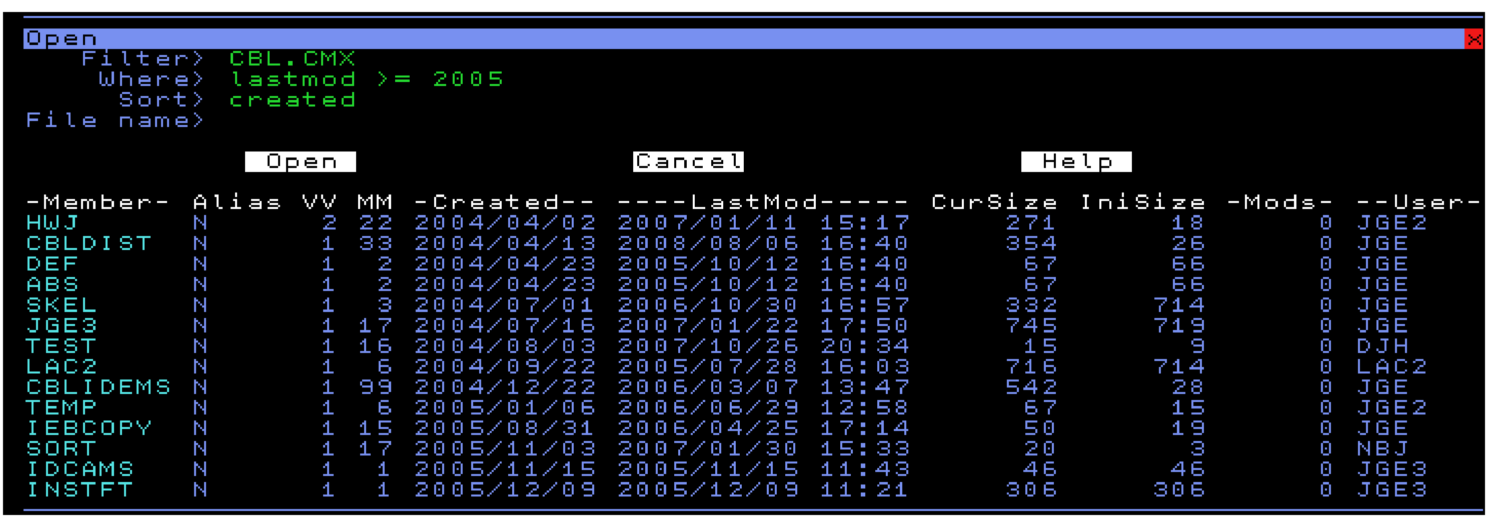

Figure 6. CBLe text edit OPEN Dialog Window.

The Open dialog allows the user to browse cataloged data set entries and PDS(E) members and select an entry for CBLe text edit. The display of entries may be scrolled up and down using PF7 and PF8 respectively.

Field Entries:

Filter>

Specifies a fileid mask filter. The filter supports the following wild cards:

(**Note:** for CMS, valid qualifiers may be considered to be File Name, Type or Mode.)

- `*` A single asterisk indicates that either a qualifier or one or more characters within a qualifier can occupy that position. An asterisk can precede or follow a set of characters.
- `**` A double asterisk indicates that zero or more qualifiers can occupy that position. A double asterisk cannot precede or follow any characters; it must be preceded or followed by either a dot or a blank.
- `%` A single percent sign indicates that exactly one character can occupy that position. (Up to 8 percent signs can be specified in each qualifier.)

On MVS or VSE, an entry field that does not contain an * (asterisk) wild card will be appended with *.** to list those cataloged data sets whose names begin with the entry string.

When the Open dialog is opened, the Filter field is populated with the fileid mask or PDS(E) library name entered the last time the OPEN window was started in the current CBLe session. If an Open window has not already been opened for the current instance of CBLe, then the fileid of the current display window is used.

Where>

Further filter the entries displayed by specifying a CBLi List window, SQL-style WHERE clause involving the fields displayed in the Open window.

```
Entry << 'TEST'  &  Vol = 'CBLM05'
```

Sort>

Sort the entries by column name by specifying a CBLi List window SORT clause.

File name>

Specifies the DSN or member name of the entry to be edited prior to selecting the Open button. Alternatively, if the required DSN or member is amongst the entries displayed in the Open list, then simply select the entry directly.

CBLe SDE Edit Dialog Window

The Edit Dialog window may be opened via the following:

- Select 'Structured edit' from the File menu in the CBLe frame window menu bar.
- Execute the SDE CLI command EDITDIALOG on the command line of an existing SDE window view, or from a CBLe text edit view if preceded by CBLi CLI command, SDATA.
- Execute the CBLi CLI command SDE with no parameters.
- Execute the prefix command "SD" from an Execute CBLVCAT or file List type window.

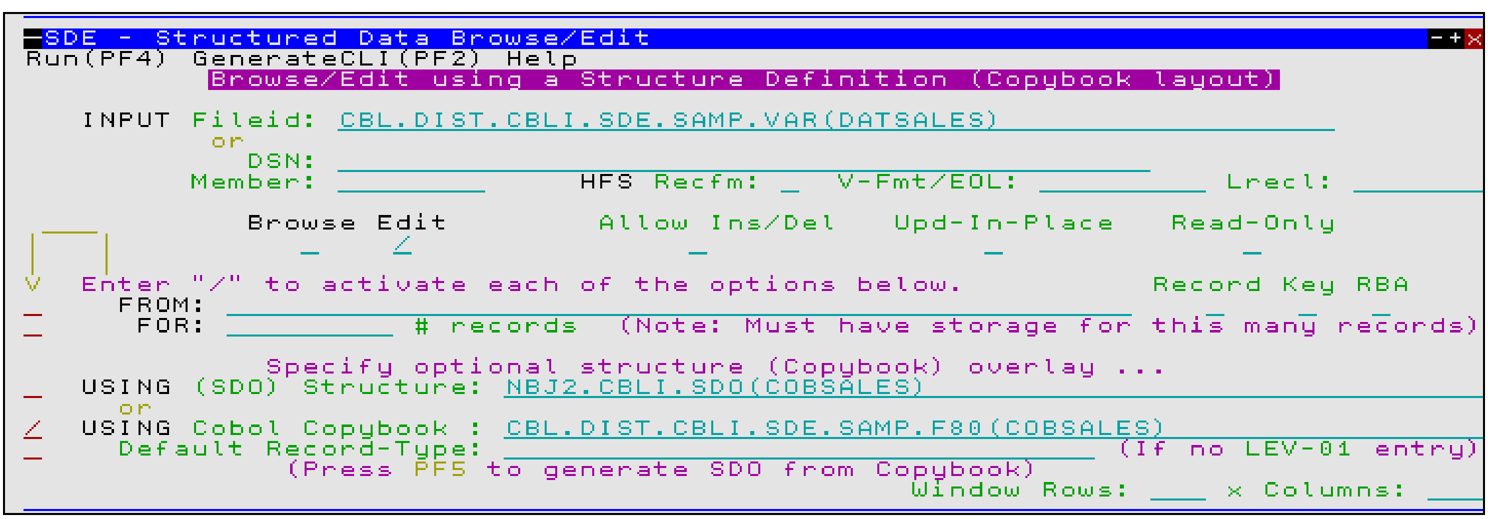

Figure 7. Structured Edit Dialog Window.

The CBL Structured Edit dialog window generates an SDE BROWSE or EDIT command to open an SDE window view in the current CBLe frame window.

Menu Items:

Run(PF4)
Select this item or hit PF4 to begin immediate execution of the SDE BROWSE/EDIT command.

GenerateCLI(PF2)
Select this item or hit PF2 to generate the SDE BROWSE/EDIT CLI command syntax for the parameters entered by the user. The generated BROWSE/EDIT command is displayed in a temporary CBLe text edit window in a format suitable for execution using CMDTEXT (PF4).

The user has the opportunity to edit the command prior to its execution and/or copying it to the home (CMX) command centre for future reference and re-execution.

GenerateCLI should be used where one or more of the fields within the SDE - Structured Data Browse/Edit window is not sufficiently large enough to contain the entered text.

Help
Open the General Help for the SDE Structure Data Browse/Edit window.

Field Entries:

By default, field entries are populated with arguments and options that were entered the last time the Structured Edit dialog window was used. The current value of each field is stored as a CBLiINI variable when the edit is actioned. These variables and subsequently saved under section SDE of the USER CBLiINI data set, when the CBLi session is ended normally.

Most field entries are optional and need to be enabled by entering "/" in the activation field which precedes the field.

`INPUT:`
Identifies the sequential or VSAM data set or PDS(E) member to be browsed or edited.

Enter the full fileid of the data set in the Fileid field **or** by enter the data set name in the DSN field and, optionally, a PDS(E) member name in the Member field.

For HFS file input, optionally specify the file's maximum record length (default 512) and Record Format or EOL delimiters. If RECFM V is specified, the EOL field is used to optionally specify the offset, length and origin parameters (default 0,2,0) which respectively define the offset and length of a binary field within the record containing the record length, and the start of the record data.

`Browse/Edit`
Select whether the data set is to be opened for browse or edit. Note that these fields are mutually exclusive.
Default is Edit.

If Edit is selected, then the type of edit to be used may also be specified as follows:

Allow Ins/Del	Perform full function edit of the data. Records may be inserted, deleted, copied, moved and the record contents altered so that, if appropriate, the record length is changed. This is default if no edit type is specified.
Upd-In-Place	Perform Update in place edit of the data. Records can not be inserted, deleted, copied or moved. Record contents may still be altered, however, record length must not change.
Read-Only	Perform full function edit of the data as detailed in "Allow Ins/Del", however, changes may not be saved unless the fileid is first updated to that of a non-existant data set. (See SDE CLI command, SET FILEID or any of the alternative SDE SET options that alter the fileid.) On saving the data, the user will be prompted to allocate/define the new data set.

Each of these edit type fields are mutually exclusive.

`FROM:`
If enabled, specifies the location within the data set from which records are to be displayed, otherwise, records are displayed from the first record in the data set. At least one of the supported options **Record**, **Key** or **RBA** must also be selected.

`Record`
Record may be selected for files of any data set organisation and indicates that the FROM field represents a record number within the data set.

`Key`
Valid only for VSAM KSDS, Key indicates that the FROM field represents a KSDS key field string. If the key is not found, the next record with the a key string greater that that specified, becomes the first record of the display.

`RBA`
Valid only for VSAM KSDS and ESDS, RBA indicates that the FROM field represents a relative byte address within the data set. The RBA location must point to the start of a record otherwise a VSAM point error will occur.

Note that, if FROM or FOR is enabled for Edit, then **Upd-In-Place** edit must be selected.

`FOR:`

If enabled, specifies the number of records to be displayable so that the End-of-Data line occurs following this number of lines.
Default is to make displayable all records following the FROM start location.

`USING (SDO) Structure:`

If enabled, specifies the SDE structure (SDO) name to be used to map record fields in the edited data set. The SDE structure must already exist, having been generated from a COBOL or PL1 copy book or using SDE internal syntax via the SDE CREATE STRUCTURE command.

If no structure is specified, each data set record will be of the default record type "Record", i.e. a single character field of length equal to that of the record.

`USING Cobol Copybook:`

If enabled, specifies the COBOL Copybook name that will be used to generate the SDE structure (SDO) necessary for the structured browse/edit.

The SDO may be generated using either of the following methods:

1. Prior to browse/edit, specify an SDO name in the **USING (SDO) Structure** field and then hit PF5 to execute the CREATE STRUCTURE CLI command which creates the SDO from the specified COBOL copybook.
   Note that, neither the **USING (SDO) Structure** nor the **USING Cobol Copybook** field need to be enabled to do this.

   Having generated the SDO, enable the **USING (SDO) Structure** field before executing the browse/edit.

2. Enable the **USING Cobol Copybook** field and specify the COBOL Copybook name. On executing the browse/edit, the dialog automatically generates a temporary (in-storage) SDO having the same name as the COBOL copybook PDS(E) library member.

   This temporary SDO is then used to browse/edit the data set.
   Note that generating a temporary SDO has a performance overhead and so, if the copybook is to be used to frequently edit data, then it is recommended that a permanent SDO is generated.

Associated COBOL Copybook field:

`Default Record-Type:`

Enable this field and specify a record type name to be used if no level 01 records exist in the COBOL copybook.

For more information on use of COBOL copybooks where no 01 level record exist, refer to CREATE STRUCTURE.

`Window Rows: x Columns:`

If not maximised, specify the dimensions of the SDE browse/edit window containing the data.

CBLe Find Dialog Window

The CBLe text edit Find dialog window may be opened via the following:

- Select **Find** from the **Edit** menu item in the CBLe Main Menu Bar.
- Enter the CBLe command FINDDIALOG at the command line of any document window.

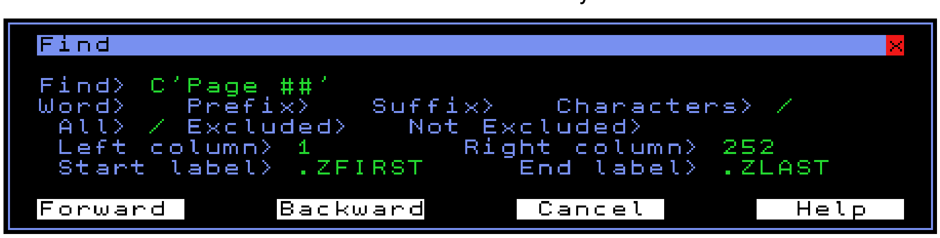

Figure 8. CBLe text edit FIND Dialog Window.

The Find dialog allows the user to find occurances of a character string in the current CBLe text edit view.

The dialog generates an ISPF edit style **FIND NEXT** or **FIND PREV** command and, for repeated execution, an **RFIND** command. Therefore, the search string entered at the **Find>** prompt may be of any format supported by ISPF FIND.

Following the search, the dialog window remains open for repeated backwards or forwards search for the string.

The position of the found search string within the file area is identified by the find cursor. The appearance of the find cursor is controlled using the SET COLOUR FCURSOR command. (Default is WHITE REVVIDEO.)
Note that FIND/RFIND will only locate a string that occurs in its entirity within the current boundaries.

When the Find dialog is opened, fields are populated with the values specified by the last FIND operation, otherwise the default values are used.

Word, **Prefix**, **Suffix** and **Characters**
These fields are mutually exclusive and define whether the search string represents:

- WORD - A word delimitted by blanks or non-alphanumeric characters.
- PREFIX - A string at the beginning of a word.
- SUFFIX - A string at the end of a word.
- CHARACTERS - A string that may be anywhere in the text. (Default)

Enter any non-blank character in one of these fields to override the existing option.

All, **Excluded** and **Not Excluded**
These fields are mutually exclusive and defines the areas of text to search.

- ALL - All lines of data (Excluded and non-excluded). (Default)
- EXCLUDED - Excluded lines only.
- NOT EXCLUDED - Non-excluded lines only.

Enter any non-blank character in one of these fields to override the existing option.

Left column and **Right column**
These fields define the first and last columns to be searched. If a value is entered in only one of the column fields or the values in both column fields are equal, then the string is only found if it starts in the specified column. Default is the left and right boundaries.

Start label and **End label**
These fields define the first and last lines of the group of lines to be searched. Default is the system assigned labels **.ZFIRST** and **.ZLAST** indicating the first and last lines of the range respectively.

CBLe Change Dialog Window

The CBLe text edit Change dialog window may be opened via the following:

- Select **Change** from the **Edit** menu item in the CBLe Main Menu Bar.
- Enter the CBLe command CHANGEDIALOG at the command line of any document window.

Figure 9. CBLe text edit CHANGE Dialog Window.

The Change dialog allows the user to find occurances of a character string in the current edit view and optionally replace them with the change string.

The dialog generates an ISPF edit style **FIND NEXT** or **FIND PREV** command and, for repeated execution, an **RFIND** command. Subsequently, if **Replace** is selected, an ISPF edit style **CHANGE .ZCSR .ZCSR NEXT** is executed, or, if **Replace All** is selected, an ISPF edit style **CHANGE ALL** is executed for the specified columns and/or group of lines.

Therefore, the search and change strings entered at the **Find>** and **Replace with>** prompts respectively, may be of any format supported by ISPF CHANGE.

Following the search, the dialog window remains open for repeated forward searches for the string.

The position of the found search string within the edit view text is identified by a find cursor. The appearance of the find cursor is controlled using the SET COLOUR FCURSOR command. (Default is WHITE REVVIDEO.)
Note that FIND/RFIND will only locate a string that occurs in its entirity within the current boundaries.

When the Change dialog is opened, fields are populated with the values specified by the last FIND operation, otherwise the default values are used.

If a replace is actioned and the length of the replace string is greater than that of the find string, then the data following is shifted to the right.

Word, **Prefix**, **Suffix** and **Characters**
These fields are mutually exclusive and define whether the search string represents:

- WORD - A word delimitted by blanks or non-alphanumeric characters.
- PREFIX - A string at the beginning of a word.
- SUFFIX - A string at the end of a word.
- CHARACTERS - A string that may be anywhere in the text. (Default)

Enter any non-blank character in one of these fields to override the existing option.

All, **Excluded** and **Not Excluded**
These fields are mutually exclusive and defines the areas of text to search.

- ALL - All lines of data (Excluded and non-excluded). (Default)
- EXCLUDED - Excluded lines only.
- NOT EXCLUDED - Non-excluded lines only.

Enter any non-blank character in one of these fields to override the existing option.

Left column and **Right column**
These fields define the first and last columns to be searched. If a value is entered in only one of the column fields or the values in both column fields are equal, then the string is only found if it starts in the specified column. Default is the left and right boundaries.

Start label and **End label**
These fields define the first and last lines of the group of lines to be searched. Default is the system assigned labels **.ZFIRST** and **.ZLAST** indicating the first and last lines of the range respectively.

CBLe Sort Dialog Window

The CBLe text edit Sort dialog window may be opened via the following:

- Select **Sort** from the **Actions** menu item in the CBLe Main Menu Bar.
- Enter the CBLe command SORTDIALOG at the command line of any document window.

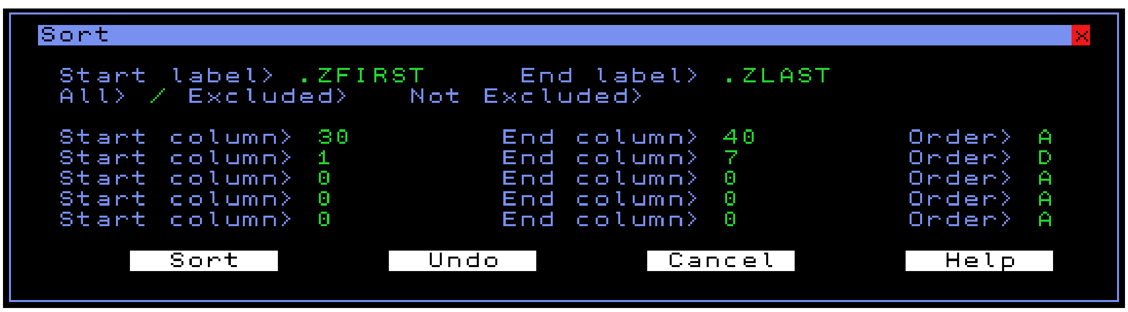

Figure 10. CBLe text edit SORT Dialog Window.

The Sort dialog allows the user to arrange data in the current edit view.

The dialog generates an ISPF edit style **SORT** command, sorting only data that falls within the current boundary settings.

Following the search, the dialog window remains open to allow further sorting.

Start label and **End label**
These fields define the first and last lines of the group of lines to be searched. Default is the system assigned labels **.ZFIRST** and **.ZLAST** indicating the first and last lines of the range respectively.

Start column and **End column**
These fields define the start and end of each sort field. A maximum of 5 sort fields may be specified that must fall within the current boundaries and must not overlap. If a start column is specified on the last sort field without an end column, then the right boundary is used by default.

Order
This field defines the order (Ascending or Descending) in which the data is to be arranged within each individual sort field.

CBLe Fill Dialog Window

The CBLe text edit Fill dialog window may be opened via the following:

- Select **Fill** from the **Actions** menu item in the CBLe Main Menu Bar.
- Enter the CBLe command FILLDIALOG at the command line of any document window.

Figure 11. CBLe text edit FILL Dialog Window.

The Fill dialog allows the user to fill a marked block with text.

The dialog generates a CBLe FILLBOX command.

Before the Fill dialog can be activated, a block of text must first be marked within the text display of an edit view. This is done using <PF15> or <PF16> which are respectively assigned to MARK BOX and MARK LINE by default.

The **Fill Char/String** field may contain a single character or a character string. If no parameter is specified, the marked block is filled with blank characters.

If a single character is specified, the character is repeated to occupy all marked positions in the text.

If a character string is specified, the string is repeated once at the rightmost column for each line of the marked block. Truncation or blank padding will occur if the length of the string is respectively greater than or less than the marked block width.

Note that the FILLBOX command is influenced by the setting of HEXSTRING and ZONE.

Editing HFS files

z/OS UNIX System Services (USS) provides a UNIX style hierarchical file system where files are organised in a directory tree structure. The z/OS hierarchical file system may comprise HFS, ZFS and NFS file system types.

Throughout CBLi documentation, these file systems are collectively referenced as the HFS file system, and entries in these systems are referenced as HFS files or HFS paths.

CBLe text edit and SDE edit views may be opened to Edit and Browse binary or text data in any HFS file.

To access HFS files for CBLe or SDE edit/browse:

- Open a List HFS Path window, using the HFS Dir Path item of the List drop-down menu or the LISTPATH command, and select an entry.
- Execute EDIT or BROWSE with a fileid which is either an absolute or relative HFS path name.
- Open the Structured Data Browse/Edit dialog window to specify an absolute or relative HFS path name.

The following CBLi functions support HFS files:

- CBLe EDIT & BROWSE.
- SDE EDIT & BROWSE - Full or In-place edit with Copy Book overlay.
- FSU - File Search/Update.
- LISTPATH - HFS Path lists including Prefix command support.
- USS Commands for Data and Environment Management. (LINK, UNLINK, MKDIR, RMDIR, CHDIR, etc.)
- ERASE
- RENAME

HFS File Records

Unlike MVS data sets which are record oriented, HFS files are byte oriented and so records within the byte stream are identified as being in one of the following formats:

- EOL (End-of-Line) character delimited. (NL, CR, LF, CRLF, LFCR, CRNL or a 2-byte, user supplied *string*.)

- Fixed length.

- Variable length. (Records include a length field at a pre-determined offset.)

The format used is determined by user supplied parameters on the EDIT, BROWSE, FSU commands or via the equivalent dialog windows.

By default, the EOL delimiter format is used with an EOL delimiter character being that stored in the file's directory entry information. If undefined, the EOL delimiter NL is used.

HFS File Name Specification

CBLi maintains the concept of the user's home directory, defined by RACF, and the **current working directory**. The current working directory may be updated using the CBLi CLI command, USS CHDIR.

An HFS file may be referenced via an **absolute** path, starting at the root directory, or a path **relative** to the current working directory. Furthermore, it is case sensitive and, if it contains special characters, blanks or commas, should be enclosed within single quotes (apostrophes) or double quotes.

CBLi imposes no restriction on the length of an HFS path name and so is subject only to those restrictions imposed by z/OS USS standards.

The **name portion** of the HFS path is the character string that follows the last "/" (slash) or, if no "/" exists, is the entire relative HFS path name. The name portion may contain wild card characters, thus allowing generic HFS file names to be specified for HFS Path Lists and FSU (File Search/Update) utilities.

By default, where specification of a fileid may either be that of an MVS data set name or an HFS path name and interpretation of the type of fileid entered is ambiguous, then an MVS data set name is assumed. To avoid ambiguity, users should include a "/" (slash) within any relative HFS path names.
e.g. `EDIT ./dev.hfs.file` references a file in the current working directory, whereas `EDIT dev.hfs.file` references the MVS QSAM file DEV.HFS.FILE.

Any relative HFS path name or symbolic link is resolved to an absolute file path and displayed in the title bar of the CBLe or SDE edit view. To conform with existing CBLe concepts, the absolute HFS path name of a file may be split into the following components:

FILEID or **DSN**	The complete absolute HFS file path name.
FPATH	The directory path from the root directory up to, but not including, the last "/" (slash) character in the fileid.
FMODE	The first level directory name above the root directory in the fileid.
FNAME or **MBR**	The character string following the last "/" (slash) and immediately preceding the last "." (dot/period) in the fileid. If no "." exists, FNAME runs to the end of the fileid.
FTYPE	The character string following the last "." (dot/period) in the fileid. If no "." exists, FTYPE is a null string.

Data Protection

User access to individual HFS files is dependent upon the file's permission bits and the configuration of OMVS settings within the user's RACF definition.

When an HFS file is opened for edit, an exclusive enqueue is established in the SPFEDIT queue which is of a format which is identical with the enqueue issued by ISPF for the edit of HFS files. The enqueue resource name is 12 bytes in length and comprsises three integers that represent the file's inode number, device number and a sysplex indicator. (See: "ISPF Planning and Customizing", "z/OS UNIX file Enqueue" for full details.)

Although the manual states that this enqueue is compatible with that issued by the z/OS UNIX OEDIT command, this has been found to be incorrect in z/OS 1.9. The sysplex indicator flag in the OEDIT enqueue resource name is set as x'20' instead of x'01' as documented. As such, OEDIT and ISPF edit do not adhere to the same file enqueuing standard.

Editing large files

CBLe always reads the whole of the file being edited into virtual storage. If a file is too big to fit in virtual storage then it cannot be edited.

However, support for edit of files that are too large to be loaded into the available region is available via the SDE (Structured Data Environment) edit feature. See the CBLe SDE Edit Dialog Window and Structured Data Environment Manual for help on starting an SDE edit view.

When a CBLe edit window is loading a file and runs out of virtual storage an error message is issued (EDT063E) and the attempted edit is aborted. It can take a long time for CBLe to exhaust storage and this time is wasted if the edit is eventually going to fail.

To give the user some control over this situation there are two CBLIINI variables which can be set in the (Edit) section of the SYSTEM or USER CBLIINI files:

SizeWarning
: The file size warning threshold. If a file is bigger than this value the user will be prompted to confirm that the file is to be edited. The default value is one megabyte.

LoadWarning
: The file load increment warning threshold. When a file is being loaded a count is kept of the number of bytes loaded. If this count excedes the LOADWARNING value then a popup message box is displayed to prompt the user to confirm that the load is to continue. The default value is one megabyte.

This feature helps with the case in MVS where the size of some files (e.g. PDS members) is not always known accurately when the file is opened.

Editing VSAM files

More flexible editing of all types of VSAM data sets including copy book overlay is available via the SDE (Structured Data Environment) edit feature. See the CBLe SDE Edit Dialog Window and Structured Data Environment Manual for help on starting an SDE edit view.

CBLe supports READ ONLY edit of most VSAM data sets. However, READ/WRITE edit is supported for VSAM data sets that have a high-used RBA of zero. This occurs when:

1. The data set is defined with parameter REUSE.
   Whenever CBLe is directed to save data to disk (SAVE, FILE, etc.), it has to re-open the data set for output. For a VSAM file defined with REUSE, CBLe re-opens the data set with the RESET attribute which resets the high-used RBA to zero.

2. Data has never been written to the data set.
   When attempting to save a new VSAM data set, the DEFINE VSAM dialog window is opened prompting the user to allocate it first.

   Beware, having saved data to an empty data set defined with NOREUSE, the high-used RBA is no longer zero and subsequent attempts to save the file will fail with the following error message:

   ```
   EDT085E VSAM PUT error for file dataset : return code x'8' reason code x'8'.
   ```

When editing KSDS data sets, care must be taken to preserve the primary key sequence. If an attempt is made to save a KSDS data set that is not in key sequence, then one of the following error messages is returned:

```
EDT086E Duplicate keys - records m and n  have the same key.
EDT087E Sequence error - record m has a higher key than record n.
```

Focus Line and Column

The concept of file focus is a very important one in CBLe. The file focus line and/or column are implied input parameters to most CBLe commands which refer to the contents of the file being edited.

For example, if you issue the DELETE command, which deletes lines from your file, the first line deleted is the focus line (more lines may be deleted, depending on the parameters supplied on the DELETE command). If you issue the CDELETE command, which deletes columns from a line in your file, the first column deleted is the focus column of the focus line (more columns may be deleted, depending on the parameters supplied on the CDELETE command).

As well as being the starting point for the effect or scope of many commands, the focus line and column may be changed as a result of executing the command. The documentation of each command will explain how the command uses and changes the focus.

The focus line and column are defined by the position of the cursor when a command is executed:

Cursor Location	Focus Line	Focus Column
In file area.	Line containing cursor.	Column containing cursor.
In prefix area.	Line containing cursor.	Left column of current file area in the current edit view window.
Not in file or prefix area. (e.g. command line)	The current line. i.e. The top line of the current edit view window.	The current column. i.e. A column number which is, by default, column 1 of the file and is changed by the commands CLOCATE, CFIRST, CLAST and SET ZONE. The current column is indicated in the scale line by a vertical bar (|).

Using Marked Blocks

A marked block is a highlighted rectangular area of text in the file you are editing which can be used as the target of many edit commands. For example you can delete, copy, move or fill a marked block. There are two types of marked block:

Line blocks	For line blocks the marked rectangle is the width of the file (current LRECL). The left edge of the marked block is column 1 and the right edge is the last column in the file.
Box blocks	For box blocks the width of the marked rectangle can be any value from 1 to the width of the file.

Both line and box blocks may span any number of contiguous lines in the file display area.

You mark a block with the MARK command. You can also use the prefix commands ML (mark line) and MB (mark box). By default function keys PF16 and PF15 are assigned to MARK LINE and MARK BOX respectively.

When you first use the MARK command the marked block is always only one line deep (the focus line) . When you issue the next MARK command, the marked block is extended (or reduced) to the current focus line.

If you issue a MARK LINE command when a box block is already marked in the current file then the box block is changed to a line block before being extended (or reduced) to the current focus line. Similarly, if you issue MARK BOX when a line block is marked it is changed to a box block.

You can unmark the current marked block with the RESET BLOCK command which by default is assigned to function key PF24.

Only one file in the ring of edited files can contain a marked block. If you mark a block in one file and a marked block already exists in another file you are editing then the existing marked block is unmarked before the new mark command takes effect.

Targets

Targets are used as parameters to a large number of CBLe commands in order to identify search criteria on data in the current file.

Successful target location can be influenced by the following SET options:

ARBCHAR, CASE, HEXSTRING, STAY, STREAM, VARBLANK, WRAP, ZONE.

Line Targets (line-target)

Line targets identify one or more lines within the edited file for which the command will be actioned.

Absolute Line Number Target

`:10`
: Target is line number 10.

Specify ":" (colon) immediately followed by an integer to indicate that the target is an absolute line number.

Relative Line Number Target

`10`
: Target line is 10 lines down from the focus line.

`-3`
: Target line is 3 lines up from the focus line.

`-*`
: Target line is line preceding the first line of the current RANGE setting.

Specify an integer to indicate that the target is a line offset from the focus line.

Alternatively, specify "*" (asterisk) to indicate the maximum offset should be applied. This will imply a line preceding the first line or following the last line of the current RANGE setting.

The direction of the offset is determined by a preceding "+" (plus) for a positive offset and "-" (minus) for a negative offset. By default, the offset is "+".

String Target

`/Hello/`
: Target line is the first line following the focus line that contains string "Hello".

`#ab/c#`
: Target line is the first line following the focus line that contains string "ab/c".

`-/Hello/`
: Target line is the first line previous to the focus line that contains string "Hello".

`/Hello /`
: Target line is the first line following the focus line that contains string "Hello" immediately followed by a blank.

`~/Hello/`
: Target line is the first line following the focus line that does not contain the string "Hello".

`-~/Hello/`
: Target line is the first line previous to the focus line that does not contain the string "Hello".

`word /ask/`

Target line is the first line following the focus line that contains the word "ask". **Note:** "asking" or "flask" will not match.

`-suffix /ion/`
Target line is the first line previous to the focus line that contains a word ending in "ion".

`/hello/ & /welcome/`
Target line is the first line following the focus line that contains both the strings "hello" and "welcome".

`-/hello/ | /goodbye/`
Target line is the first line previous to the focus line that contains either the string "hello" or "goodbye".

A string target is a sequence of characters enclosed in delimiters. CBLe will scan one line at a time starting at the line following the focus line for a forward scan, or the line preceding the focus line for a backward scan.

On encountering bottom of file (or top of file for a backward scan), the WRAP setting will determine whether the scan will continue on lines at the opposite extreme of the file.

The "/" (slash) character is normally used as the delimiter character. However, any non-alphanumeric character that does not have a special meaning to CBLe may be used as a string delimiter (e.g. "%", "#", "."). The delimiter character used must not appear in the target string itself.

Leading and trailing blanks contained in target strings are respected.

"¬" (not), "^" (carat) or "~" (tilde) preceding the target string may be specified to indicate scan for a line that does not include the target string.

The string target may be prefixed by one of the following special keywords:

`Word`	The target string must match a complete word.
`Prefix`	The target string must match the leading characters of a complete word.
`Suffix`	The target string must match the trailing characters of a complete word.

A word is considered to be a string of characters with the dlimiter characters blank or any non-alphanumeric character.

String targets may be combined to form an expression using logical operators. The "and" operator is represented by "&" (ampersand) and the "or" operator is represented by "|" (vertical bar).

The direction of the scan is determined by a preceding "+" (plus) for a forward scan and "-" (minus) for a backward scan. By default, the direction is "+".

Note: , for expressions comprising multiple string targets, direction may only be specified once on the first string target.

Named Line Target

`.curr`
Target line is the line previously named ".curr" via a SET POINT command.

A named line target is a line that has been named using the SET POINT command.

Specify "." (dot) immediately followed by the line name to indicate that the target is a named line target.

Line Class Target

`blank`
Target line is the first line following the focus line that is completely blank.

`-altered`
Target line is the first line previous to the focus line that has been altered (updated or added) during the current CBLe session.

`~new`
Target line is the first line following the focus line that has not been added during the current CBLe session.

The following line class targets are supported:

`BLAnk`	Lines that are blank.
`TAGged`	Lines that have been tagged using the TAG command.
`CHAnged`	Lines that have been updated.
`NEW`	Lines that have been added.
`ALTered`	Lines that have been added or updated.
`SELECT n`	Lines that have selection level n or lines that have selection level between n and m. See SET SELECT for description of selection levels.

Column Targets (column-target)

`:66`
: Target column is column number 66.

`10`
: Target column is 10 columns to the right of the focus column.

`-3`
: Target column is 3 columns to the left of the focus column.

`-*`
: Target column is the column to the left of the current ZONE setting.

`/abc /`
: Target column is the first column following the focus column that contains string "abc" immediately followed by a blank.

`-/John/`
: Target column is the first column previous to the focus column that contains string "John".

`word /just/`
: Target column is the first column following the focus column that contains the word "just".
Note: "adjust" or "justify" will not match.

Column targets identify a column number within the edited file for which the command will be actioned. CBLe commands requiring the column-target parameter are CLOCATE and CDELETE.

The column-target may be an absolute column number, relative column number or a string target.

An absolute column number is indicated using a ":" (colon) immediately followed by the column number.

A relative column number acts in the same way as a relative line nember except that the specified integer indicates an offset to the right or left of the focus column for positive or negative offsets respectively.

Similarly, specifying "*" (asterisk) as the offset value implies a column preceding the first column or following the last column of the current ZONE setting.

String target syntax is supported in same way as for line targets. CBLe will scan one column at a time starting at the column to the right of the focus line for a forward scan, or the column to the left of the focus line for a backward scan.

For string targets only, the SET STREAM command determines whether the scan will continue over multiple lines should the target string not be found on the focus line.

Group Targets (group-target)

`:26`
: Target lines are the focus line and all lines up to, but not including line number 26.

`-8`
: Target lines are the focus line and the 7 lines immediately preceding it.

`*`
: Target lines are the focus line and all lines that follow it.

`/dummy/`
: Target lines are the focus line and all lines up to, but not including, the first line following the focus line to contain the string "dummy".

`ALL`
: Target lines are all lines in the file.

`BLOCK`
: Target is all characters in the currently marked line or box block.

Group targets identify a group of lines (target area) for which the command will be actioned. A group-target parameter is required for CBLe commands such as CHANGE, UPPERCASE, LOWERCASE and DELETE.

The focus line is treated as the first line of the group. Any form of the line-target syntax is used to flag the end of the target area, however, the target line itself is not included as part of the target area.

The group target may also be one of the following special keywords:

`ALL`	The group-target is all lines in the file.
`BLOCK`	The group-target is the currently marked line or box block.

CBLe Environment Variables

CBLe edit views support a set of system determined and user defined environment variables that may be used in line commands and macros executed form CBLe edit views.

The types of variables supported and they way in which they may be utilised by the user are documented in this section.

Variable Types

CBLe environment variables may be one of the following four types:

1. User defined environment variables set via the CBLe CLI EDITV command. Use the REXX macro, EQU, for a simple method of setting user environment variables.
2. Standard environment variables, as follow:

VarName	Description
user	User name as reported by 'Query USERname'
datetime **dtme**	Current date and time in format 'yyyy/mm/dd hh.mm'. e.g. 2006/08/09 23.59
timestmp	Current date and time in format 'yyyymmddhhmnss'. e.g. 20060809235942
date	Current date in format 'yyyy/mm/dd'. e.g. 2006/08/09
yyyy	Current 4 digit year. e.g. 2006
yy	Current 2 digit year. e.g. 06
mm	Current 2 digit month. e.g. 08
dd	Current 2 digit day of month. e.g. 09
ddd	Current 3 digit day of year. e.g. 221
time	Current time in format 'hh:mm:ss'. e.g. 23:59:42
tme	Current time in format 'hh.mm'. e.g. 23:59
tsoprefix **tsopfx**	For MVS, the defined TSO prefix. In many TSO environments, this has the same value as the %USER% userid environment variable.
hh	Current 2 digit hour of day. e.g. 23
mn	Current 2 digit minute of hour. e.g. 59
ss	Current 2 digit second of minute. e.g. 42
fi **fid**	Current file's fileid as reported by 'Query FIleid'
fm	Current file's filemode as reported by 'Query FMode'
fp	Current file's filepath as reported by 'Query FPath'
fn	Current file's filename as reported by 'Query FName'
ft	Current file's filetype as reported by 'Query FType'
ds **dsn**	Current file's DSname as reported by 'Query DSn'

3. MVS system symbols. e.g. &SYSNAME.
4. INI variables that have been explicitly set in the SYSTEM or USER CBLiINI files.

   These types of variables have the form "SYSTEM.section.varname" and "USER.section.varname" where:

SYSTEM USER	Variable as defined in the SYSTEM or USER CBLiINI file.
section	Section within the CBLiINI file. e.g. SYSTEM, EDIT, SELCOPY, RACF, etc.
varname	Name of the variable applicable to the relevant section of the CBLiINI file. e.g. InitialSize, CmdText, ProgramName, etc.

See CBLiINI Help for all standard CBLiINI variables that are significant to the CBLi program. e.g. `SYSTEM.CBLVCAT.SVC`, `USER.EDIT.SizeWarning`, `SYSTEM.HELP.DefaultPath`

In addition to the standard CBLiINI variables, user defined variables may also be entered in the SYSTEM and USER CBLiINI files and referenced in the same way as the standard CBLiINI variables. User defined CBliINI variables may be inserted using either of the following methods:

- Execute the CBLe CLI command, SET INIVAR. The variable is set with immediate effect and is automatically inserted in the USER CBLiINI file when the CBLi session is ended normally.
- Manually edit and update the relevant (USER or SYSTEM) CBLiINI file. The alterations are not immediate and will only take effect the next time CBLi is started.

Variable Substitution

CBLi supports use, and subsequent translation, of CBLe environment variables referenced in CBLe REXX macros or specified within any command string that is executed from a CBLe edit view. This includes:

1. Commands in an edited file (typically a CMX file) that are executed using the CMDTEXT facility.
2. Commands executed from the CBLe edit view command prompt.

The CBLe CLI command, SET ENVVAR, switches CBLe variable substitution ON or OFF, also defines the variable delimiter character. By default, variable substitution is switched on with '%' (percent - X'6C') as the variable delimiter character.

By default, the CBLe edit variables, **user** and **system.edit.macropath**, will be translated in the following command.

```
set MACROPath   %user%.CBLE.MACROS   %system.edit.macropath%
```

The following example of concatenated command strings could be saved in your command (CMX) file for execution via CMDTEXT (PF4), to output a lists of library members updated by your userid today. Note that ';' (semi-colon - X'5E') is the command separator character.

```
<LL %user%.cbli.cble ; WHERE USER = %user% & LASTMOD => '%date%'   \
;ll %user%.cbli.cmx  ; WHERE USER = %user% & LASTMOD => '%date%'
```

The prevailing SET ENVVAR status (ON/OFF) may be temporarily overridden by prefixing the command with VIgnore or VRespect. VIgnore bypasses variable translation and VRespect performs variable translation regardless of the SET ENVVAR status.

The following command could be issued from a CBLe REXX macro to temporarily bypass variable translation when ENVVAR is ON. (Upper case characters in the keywords indicate minimum abbreviation.)

```
VIgnore Input '<ld %&SYSNAME%.Z16.**   | List system data sets.'
```

The LISTDATASET command string, beginning '<ld ', will be inserted following the focus line of the edited data with %&SYSNAME% unchanged.

ISPF Edit Interface

The CBLe text editor can execute in one of two operating interface modes namely, CBLe and ISPF.

- The CBLe interface is the classic CBLe operational mode that is based on IBM's CMS XEDIT and Mansfield's PC Kedit.
- The ISPF interface is an operational mode based on IBM's MVS ISPF Edit.

Unless otherwise stated, features of the CBLe editor documented in this manual are equally applicable to edit views running in either operating interface mode.

This section of the manual deals specifically with features supported by the CBLe ISPF operating interface and the contrasts between the two operating interface modes.

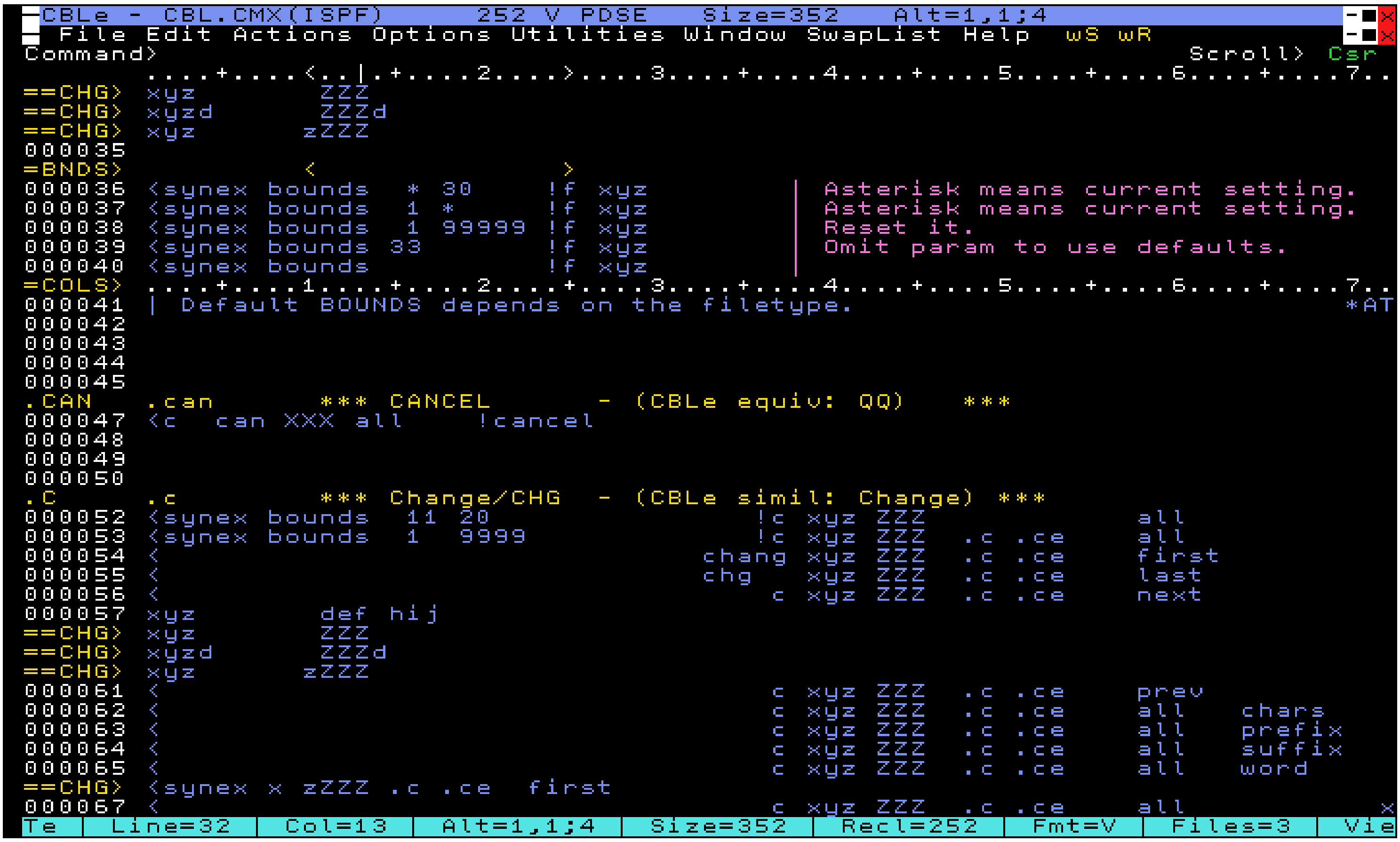

Figure 12. CBLe Text Edit View running ISPF interface mode.

ISPF Edit Features

Wherever possible, the CBLe ISPF Interface is intended to mirror functionality included in the ISPF editor.

The functionality provided by ISPF features are documented fully in IBM manuals "ISPF: Edit and Edit Macros", "ISPF: User's Guide Vol 1" and "ISPF: User's Guide Vol 2".
Although not all features of ISPF Edit are included, the CBLe ISPF interface supports the following:

ISPF and ISPF Edit Primary Commands

AUTOSAVE	**COPY**	**HEX**	**RFIND**
BNDS	**CREATE**	**HIDE**	**RIGHT**
BOTTOM	**DELETE**	**LEFT**	**SORT**
BOUNDS	**DOWN**	**LOCATE**	**TOP**
CANCEL	**END**	**MOVE**	**UP**
CAPS	**EXCLUDE**	**RCHANGE**	**X**
CHANGE	**FIND**	**REPLACE**	
CHG	**FLIP**	**RESET**	

See ISPF/CBLe CLI Command Precedence for co-existance of ISPF and CBLe commands.

ISPF Edit Line Commands

(	**BNDS**	**I**	**R**
((	**BOU**	**L**	**RR**
)	**BOUND**	**LC**	**S**
))	**BOUNDS**	**LCC**	**TF**
<	**C**	**LCLC**	**TS**
<<	**CC**	**M**	**UC**
>	**COL**	**MM**	**UCC**
>>	**COLS**	**MASK**	**UCUC**
A	**D**	**O**	**X**
B	**DD**	**OO**	**XX**
BND	**F**	**R**	

In addition to these standard ISPF line commands, the ISPF interface also supports the following CBLe prefix area commands:

"	/	**MB**	**SCALE**
""	**HEX**	**ML**	**SJ**

ISPF Edit Display Fields

- The ISPF edit enterable SCROLL field with valid entries: 0-9999, CURSOR, DATA, HALF, MAX and PAGE.
- Line labels.
  Note that the SETPT REXX macro may be used to set multiple labels corresponding to line markers in the file's text.
- Line command (prefix) area flagged text for changed lines (==CHG>), error lines (==ERR>) and special lines for boundary definition (=BNDS>) and column identification (=COLS>).

ISPF PFKey/command line concatenation

The contents of the edit view command line are concatenated to the PFKey definition. The result is executed as a single command.

As for ISPF, the caveat exists whereby the PFKey definition is a scroll command and the concatenation of the command line contents results in an invalid scroll command. In this case, if the word following the PFKey scroll command definition begins with a non-numerical character, the contents of the command line are executed prior to the PFKey scroll function.

ISPF Interface Initialisation

The prevailing operational interface mode is defined via the following methods:

1. The CBLiINI variable EDIT.INTERFACE may be set in either the SYSTEM or USER CBLiINI file. e.g.

   ```
   (Edit)
      Interface=ISPF        * ISPF or CBLe.
   ```

   This method defines the default interface for any new invocation of the CBLe editor.

   If the CBLiINI variable EDIT.INTERFACE is not set, then INTERFACE=ISPF is the default on MVS type systems, whereas INTERFACE=CBLe remains the default for VM/CMS and VSE.

2. The CBLe CLI command SET INTERFACE, allows the user to define the interface type to be used at the specified level. Supported levels are:

VIEW	The current edit view only.
FILE	All new and existing edit views of the current file.
GLOBAL	All new and existing edit views.

The parameter **INItialise** may also be specified on SET INTERFACE to intitialise the edit view(s). This redefines the PFKeys to be the defaults for the specified interface and also updates the appearance of the edit view (e.g. SCROLL field on for ISPF). e.g.

```
SET INTERFACE ISPF VIEW INI
```

ISPF/CBLe CLI Command Precedence

The set of ISPF interface primary commands are supported in addition to the existing CBLe CLI commands.

Where a command verb is associated with only one of the command interfaces (INTERFACE=CBLE or INTERFACE=ISPF), then it may be used with either interface without conflict. e.g. The CBLe CLI command, ALL, may be executed when in INTERFACE=ISPF.

The following are commands or SET options that exist for both command interfaces.

AUTOSAVE	**CREATE**	**LOCATE**	**TOP**
BOTTOM	**DOWN**	**MOVE**	**UP**
CANCEL	**FIND**	**REPLACE**	**X**
CHANGE	**HEX**	**RESET**	
COPY	**LEFT**	**RIGHT**	

When executing one of these commands, then, by default, the function and syntax of the command will be that associated with the active command interface. e.g. If INTERFACE=ISPF is active, the ISPF primary command, CHANGE, would be executed instead of the CBLe CLI command, CHANGE.

The active command interface may be temporarily overridden by prefixing the command with ICommand or ECommand. ICommand passes the command to the ISPF command interface, ECommand to the CBLe command interface.

The following command will use the CBLe version of the CHANGE command when INTERFACE=ISPF is active. (Upper case characters in the keywords indicate minimum abbreviation.)

```
ECommand Change /ABC/DEF/ * *
```

CMX (CoMmand eXecution) Files

Also referred to as command centres, CMX files are non-executable, plain text files that contain groups of associated TSO, ISPF, CBLi and CBLe commands and comment data. These groups of commands may be used to perform regular tasks and procedures that would ordinarily be issued from a command prompt.

Execution of commands within a CMX file is achieved using CBLe's CMDTEXT facility. Simply place the cursor on the line of text containing the required command (usually a line starting with '<' to signify immediate execution) and hitting the PF4 key (assigned to CMDTEXT by default.) e.g.

```
<edit     DEV.XG80.JGE001.COB(XCC532)       | Edit COBOL source.
<submit   DEV.XG80.JGE001.JCL(XCC)          | Compile COBOL source.
<browse   DEV.XG80.JGE001.LST.XCC           | Browse SYSPRINT output.
```

Note:
Use of CMDTEXT is not restricted to files of type CMX. Commands may be stored, and subsequently executed, from any editable plain text file. (e.g. REXX procedures and SELCOPY, Assembler, C/C++, COBOL source files.)

Benefits of CMX files include:

- Productivity. Faster than using menus and dialog panels or re-typing commands from scratch.
- Session. Launch edit of frequently used files.
- Management. Single point of reference for related commands and procedures.
- Environment. Set and query the local operating environment options.
- Program Development. Submit jobs, display output and invoke debuggers.
- Education. Users can keep a record of supported command syntax and working examples.

The first time a user starts CBLi following initial install, the FIRSTUSE procedure is executed to establish the user's individual working environment (User CBLiINI) and defualt command centre (CMX) files. During this process the user will be prompted to allocate these 2 new files.

The user's CMX file, also referred to as the user's HOME file, is generated from the FIRSTUSE skeleton CMX file distributed by CBL and is automatically opened for edit each time the user starts CBLi. This is the user's personal springboard into CBLi's functions and it is intended that the user add his or her own commands and comments.

This file also acts as a basic tutorial for functions provided by the current release of CBLi and it is recommended that the user take some time to read the comments and execute some of the commands in order to become familiar with CBLi's features and operation. Following an upgrade of CBLi, the user's CMX file may be automatically updated to reflect new features added to the product.

REXX Macros

User macros may be written to perform functions within a CBLe text edit or SDE edit window using the REXX procedure language.

The name associated with the CBLe text edit environment is **CBLEDIT**, whereas the equivalent name associated with the SDE environment is **CBLSDATA**.

The appropriate name should be specified on the REXX instruction, ADDRESS, if commands within the macro are to be directed to a particular edit environment. If a macro is executed from within a CBLe text edit document window, CBLEDIT is automatically set as the default environment. Similarly, CBLSDATA is automatically set as the default environment if a macro is executed from within an SDE edit document window.

The current environment may be identified using the REXX built-in function ADDRESS().

A number of REXX macros are supplied with the CBLi product bundle. A detailed description on the use of each macro is documented within the macro executable file.

BLOCK	Scan for next or previous highlighter line and make it the focus line. (Used for default CBLe window class keys PF05 and PF06)
BOX	Restrict operation of a CBLe command/macro to the ZONE and RANGE limits determined by the marked block.
BOXSEQ	Insert a numbers in sequence, one in each line of a marked block.
BOXTOT	Display the sum of all numeric values in a marked block.
CBLIINI	Display the fileid of the SYSTEM and USER CBLIINI files and the CBLIINI variables contained within.
CBLITU	Check current user is a trusted user. Executed once only by PROFIRST.
CBLIZAPL	Report all zaps that are applied to the CBLi module being executed.
CCDATE	Output today's date at the cursor in the format ccyy/mm/dd.
CMDFUNC	CBLe utility functions for dataset point-and-shoot from: 1. MVS JCL DD, IDCAMS DEFINE and TSO/CBLe ALLOC statements. 2. Any file where the cursor is positioned on a DSN. Format: `CMDFUNC   X | K | R` Where: **X** Edit a file. (Executed immediately) **K** Erase a file. (Command put on command line) **R** Rename a file. (Command put on command line) Ideally, where supported by the 3270 emulator, keyboard macros set on Ctrl-X/K/R should be created to issue the releveant CMDFUNC command.
CMDX	Execute CMDTEXT for all commands in the currently displayed text between the focus line and the specified line target.
CMEN	Generate a temporary CMX file containing all retrievable commands.
CMXAMS	Convert IDCAMS DEFINE input to CBLi AMS command format for execution using CMDTEXT.
COLSET	Set colour scheme for the current display window.
CURALL	Generate ALL command for text at the cursor position.
CURSCALE	Use SET RESERVE to insert a scale line at the cursor position.
CUT	Simulate ISPF Edit primary command, CUT.
DELALL	Delete lines from the current file that match the specified line-target.
DELNOT	Delete lines from the current file that do not contain the specified string.
DIR	Generate edited list of library entries belonging to multiple libraries suitable for CMDFUNC execution.
DIRCMD	Convert DIR macro output or CBLi command FL, LL, LC, LD Edit output to CMX file format.
DIRSORT	SORT DIR macro output.
EINI	Edit the SYSTEM (Site) or USER CBLiINI file.
EM	Edit the first macro in the macro path with the specified file name.
EQU	Set a CBLe user environment variable using EDITV.
ERA	Erase the file in the current display window.
ERASEALL	Create a CMX file containing an ERASE command for all files matching a supplied mask.
FILES	Create a temporary CMX file containinf EDIT commands for all files edited in this CBLe session.

FILESADD	Add the current fileid to the %FILES% environment variable. (Used by the FILES macro.)
FIRSTUSE	Install macro. Executed automatically for first ever logon to CBLi by a user to initialise user's environment.
FSX	Search multiple MVS PDS(E) libraries for a specified search string.
GETAVRL	Get the average record length of the specified file. (Uses SELCOPY)
HD	Insert a CBL style header line in line 1 of the current file. (Suitable for SV macro)
IEX	For CBLi on MVS ISPF only, open a split screen contianing ISPF panel nominated by an ISPF FastPath.
JCLCMX	Opens a temporary CMX file containing ALLOCATE commands generated from DD statements in an MVS batch job.
JEM	For CBLi on MVS ISPF only, sample CBLi interface to a JEM JCL validation ISPF-Edit macro.
JOBCARD	Insert a skeleton MVS jobcard in line 1 of the current file.
KBALL	Used by 3270 emulator keyboard macro (Alt-9) to redisplay all excluded lines.
KBASTTOG	Used by 3270 emulator keyboard macro (Alt-H) to toggle double-asterisk in columns 71-72.
KBBOF	Used by 3270 emulator keyboard macro (Ctl-F8) to display the last page of text.
KBCCDATE	Used by 3270 emulator keyboard macro (Ctl-F3) to put today's date in yyyy/mm/dd format at the cursor position.
KBCMDFK	Used by 3270 emulator keyboard macro (Ctl-K) to execute CMDFUNC K at the cursor position.
KBCMDFR	Used by 3270 emulator keyboard macro (Ctl-R) to execute CMDFUNC R at the cursor position.
KBCMDFX	Used by 3270 emulator keyboard macro (Ctl-X) to execute CMDFUNC X at the cursor position.
KBCMDTE	Used by 3270 emulator keyboard macro (Alt-F4) to execute command at cursor (CMDTEXT) with action of '<' reversed.
KBCURALL	Used by 3270 emulator keyboard macro (Ctl-A) to generate CBLe ALL command for text at cursor position.
KBCURH	Used by 3270 emulator keyboard macro (Ctl-Home) to execute CBLe CURSOR HOME command.
KBCURSCL	Used by 3270 emulator keyboard macro (Alt-S) to define a reserved line containing a scale starting at cursor position.
KBDELW	Used by 3270 emulator keyboard macro (Ctl-W) to delete characters at the cursor position up to the end of the word.
KBDELWDM	Used by 3270 emulator keyboard macro (Alt-W) to delete the word at the cursor position.
KBDOWN1	Used by 3270 emulator keyboard macro (Ctl-DownArrow) to place focus on the line following the current focus line.
KBEDTADD	Used by 3270 emulator keyboard macro (Alt-A) to add a line following the focus line.
KBEDTDEL	Used by 3270 emulator keyboard macro (Alt-D) to delete the focus line.
KBEDTDUP	Used by 3270 emulator keyboard macro (Alt-2) to duplicate the focus line.
KBEDTSJ	Used by 3270 emulator keyboard macro (Alt-J) to split or join a line at the focus column.
KBEXCLD	Used by 3270 emulator keyboard macro (Alt-X) to exclude the focus line or marked block.
KBFILL	Used by 3270 emulator keyboard macro (Alt-F) to fill a marked block with specified character(s).
KBHDSUB	Used by 3270 emulator keyboard macro (Alt-8) to convert focus line to a formatted (triple asterisk) header line.
KBHILO	Used by 3270 emulator keyboard macro (Alt-7) to toggle between triple/double asterisk on focus line.
KBMAJDN	Used by 3270 emulator keyboard macro (Ctl-F6) to make the first line following the current line that contains a triple asterisk/equals, the current line.
KBMAJUP	Used by 3270 emulator keyboard macro (Ctl-F5) to make the first line previous to the current line that contains a triple asterisk/equals, the current line.
KBMARKL	Used by 3270 emulator keyboard macro (Alt-L) to mark the focus line.
KBMCURR	Used by 3270 emulator keyboard macro (Ctl-/) to make the focus line the current line.
KBMINDN	Used by 3270 emulator keyboard macro (Alt-F6) to make the first line following the current line that contains a double asterisk/equals, the current line.
KBMINUP	Used by 3270 emulator keyboard macro (Alt-F5) to make the first line previous to the current line that contains a double asterisk/equals, the current line.
KBPFXTOG	Used by 3270 emulator keyboard macro (Alt-N) to toggle the prefix area on and off.
KBRESET	Used by 3270 emulator keyboard macro (Alt-U) to unmark a block.
KBSV	Used by 3270 emulator keyboard macro (Ctl-S) to issue the SV macro to update the timestamp in header and save the file.
KBTAGTOG	Used by 3270 emulator keyboard macro (Ctl-H) to toggle highlighting of the focus line.

KBTOF	Used by 3270 emulator keyboard macro (Ctl-F7) to display the first page of text.
KBTRB	Used by 3270 emulator keyboard macro (Alt-T) to insert TRACE R/TRACE OFF REXX instructions around a marked block of text. (For REXX exec/macro development.)
KBTRBOFF	Used by 3270 emulator keyboard macro (Ctl-T) to remove TRACE instructions inserted by KBTRB.
KBUP1	Used by 3270 emulator keyboard macro (Ctl-UpArrow) to place focus on the line before the current focus line.
KBWINXR	Used by 3270 emulator keyboard macro (Alt-R) to restore window position/size from a saved value.
KBWMAXFR	Used by 3270 emulator keyboard macro (Ctl-Num+) to maximise the current MDI frame window.
KBWRESFR	Used by 3270 emulator keyboard macro (Ctl-Num-) to minimise the current MDI frame window.
LDIFF	Compares text in the current file with text in the next file in the ring and highlights the 1st difference found following the current line.
LLX	List members from multiple MVS PDS(E) libraries.
LM	Open a List Library window, one for each library in the macro path.
LVX	List data sets entries from multiple MVS DASD VTOCs.
MDL	Set current window's size to dimensions imposed by standard terminal hardware models.
MI	SET CASE M I prior to executing the supplied command and reset it when the command completes.
NAM	Place the current file's fileid on the command line for subsequent editing.
MOVEBLKR	Move a marked block and reset instead of leaving it marked. (PF18)
MR	SET CASE M R R prior to executing the supplied command and reset it when the command completes.
ONLY	For CBLe and SDE windows, display only lines that match the specified FIND search string and parameters.
OP	For CBLi on MVS ISPF only, gives quick access to the SDSF Operator System Log (LOG) panel and optionally execute an operator command.
OQ	For CBLi on MVS ISPF only, gives quick access to the SDSF Output Queue (ST Status of jobs) panel.
PASTE	Simulate ISPF Edit primary command, PASTE.
PROFILE	The CBL default PROFILE macro.
PROFIRST	Called by PROFILE macro (once only) to add useful edit buttons to the MDI frame window's menu bar.
PROSITE	Called by PROFILE macro (once only) after PROFIRST to apply Site wide overrides to default CBLE settings.
PROUSER	Called by PROFILE macro (once only) after PROSITE to apply User overrides to default CBLE and PROSITE macro settings.
QX	Display REXX stem variables returned by CBLe command EXTRACT for the specified extract option.
RENAME	For CBLi on MVS only, intercept CBLe RENAME and check for current fileid removing the ENQ if necessary.
RESTREAM	Re-Stream a file UNSTREAMed from a VSE LIBR RECFM=S member.
RINGL	Display the current ring of files as a popup menu. A file may be selected from the menu and made the current file. C|U option displays changed files only.
RST	Re-edit the current file discarding all unsaved changes.
SDBPOPUP	Display the SELCOPY Interactive Popup menu for the focus operation.
SDBTRACK	Performs SELCOPY Interactive command TRACK and colour the tracked expression in all edit views.
SDBWINX	Save and restore the size and locations of all edit view windows in the SELCOPY Interactive MDI frame window.
SDECOB	Structured Data Environment - Generate a COBOL copybook from SDE structure.
SDECOPYF	Structured Data Environment - Copy data in selected columns to new data set.
SDEPROF	Structured Data Environment - The CBL default SDE Edit profile macro.
SDERTALL	Structured Data Environment - Issue a command against all available record types.
SDESEL	Structured Data Environment - Generate SELECT command for all columns in the default record type.
SDEZOOMW	Structured Data Environment - Open a new Edit view containing the ZOOMed record.
SETPT	Scans the file for strings beginning with "." within specified zone columns and uses SET POINT to name the line.
SHOW	Un-exclude specified number of excluded lines.
SV	Save the current file if it has alterations and update the level in the CBL style header line. (See macro HD)
TRA	Capture TRACE output from a REXX procedure.
TRB	Insert REXX TRACE commands around a marked block of REXX statements in a macro file.
TSOC	Execute the specified TSO command then store and display any TSO messages returned in a temporary file.

UNNUM	Simulate ISPF Edit primary command, UNNUM.
UNSTREAM	Convert a VSE LIBR RECFM=S character file to RECFM=F.
VSECBLN	Insert CBLNAME ASSEMBLE and tailor for VSE specific variables. (VSE CBL product install from CMS.)
VSEINCL	Replaces 'INCLUDE module' records with the data from 'module TEXT *'. (VSE CBL product install from CMS.)
VSESITEV	Update Site-dependent variables in various VSE .Z skeleton JCL decks. (VSE CBL product install from CMS.)
VSESNAM	Insert SELCOPY NAM and tailor for VSE specific variables. (VSE CBL product install from CMS.)
WINX	Save and Restore the size and location of the current edit view within the MDI frame window.
WW	Start a new CBLe or SDE window view of the current file and optionally execute a command.

These, and all other macros that are to be made accessible to the CBLe session, must exist within a library referenced by the macro path.

The macro path is a list of directories assigned to the **Edit.MacroPath** variable which may be set via the System or User CBLIINI file or via the CBLe SET MACROPATH command. The macro path is searched in order until a file name that matches the macro name is found.

If no macro path is defined then the following occurs:

- For MVS systems, no action is taken. In order to execute CBLe macros a macro path is mandatory.
- For CMS systems, CBLe will search for a matching file name with a file type of **CBLE**, **CBLEDIT** or **XEDIT** (in order) on each accessed minidisk.
  Note that macro ABC.XEDIT.A will be found before ABC.CBLE.B.
- For VSE systems, CBLe will search the LIBDEF PROC library SEARCH chain for a matching member name.

If CBLe is unable to locate a CBLe macro then use CBLe command QUERY MACROPATH to verify your macro path, otherwise, please contact your Systems Programmer.

Users should refer to IBM REXX documentation for assistance when writing edit macros in the REXX language. CBL supplied macros are also a good starting reference for examples of REXX procedures that use CBLe commands.

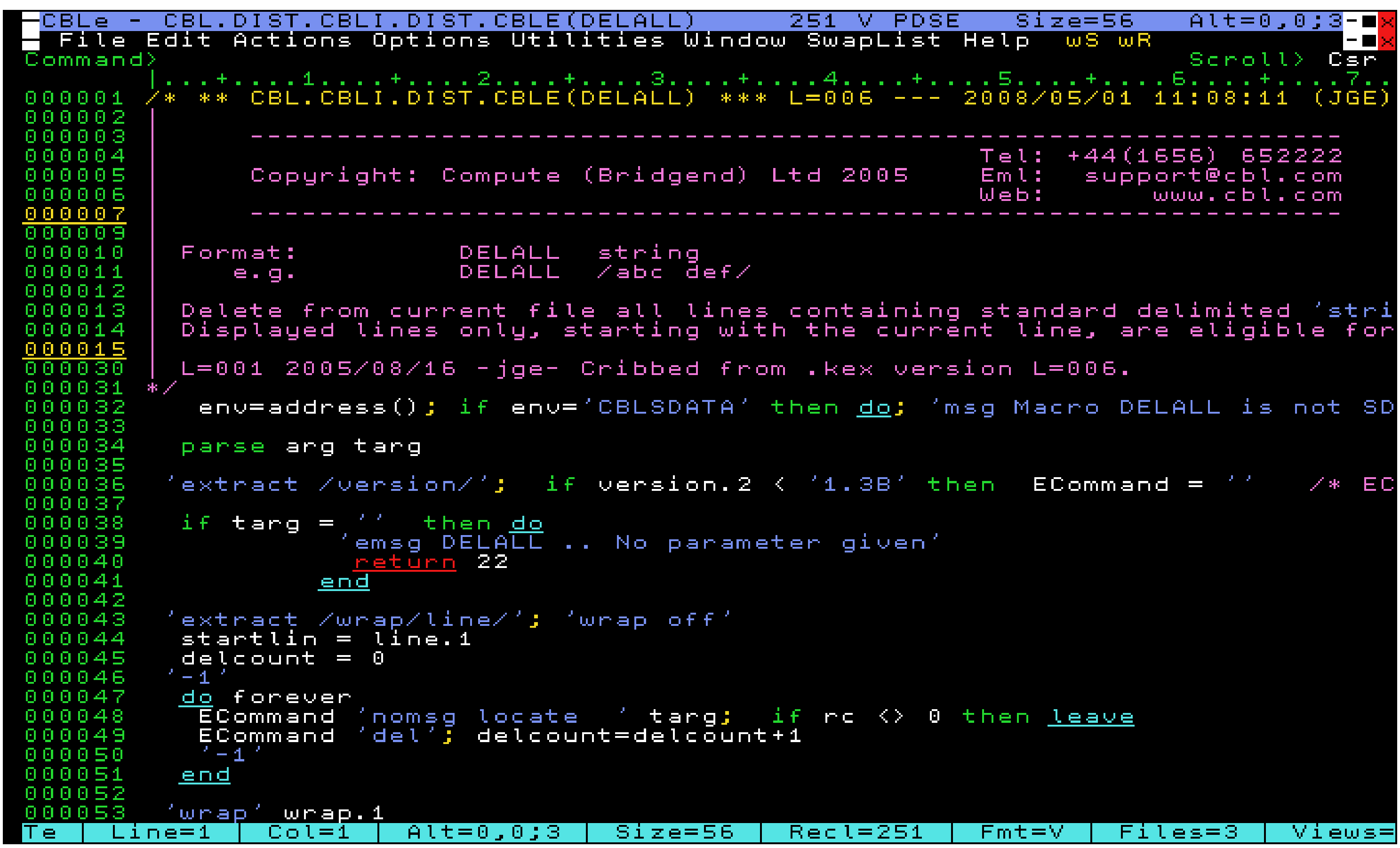

Figure 13. CBLe REXX edit macro - DELALL

CBLe PROFILE Macro

When a new file is opened, CBLe searches the macro path libraries for the first occurrence of the profile macro name as defined in the CBLiINI file by variable **Edit.DefProfile**. If undefined, CBLe uses a profile macro name of **PROFILE**.

The profile macro may be used to define the user's CBLe environment. Any System or User CBLIINI file options that correspond to CBLe SET command options, may be overridden for a CBLe session by including the SET command in the PROFILE macro.

A default profile macro, PROFILE, is distributed with the CBLi product bundle. This macro, amongst other things, sets different colours for each file in the ring of files being edited and also tags any lines containing two or more asterisks. These settings are CBL preferences and may be changed.

Command Line Commands

CBLe commands may be issued from:

1. The CBLe command line.
2. A text file using the CMDTEXT facility.
3. A CBLe macro.
4. A programmable function key.

Multiple commands may be issued in a single invocation by separating each command with the special line end character as defined by SET LINEND (set to "!" exclamation mark by default.)

Note that, if LINEND is ON and the user wants to temporarily switch it off in order to execute a command that contains the linend character as text, then this may be achieved by prefixing the command with the line end character. e.g.

`!c/Hello there!/Goodbye/  10 1`
: Change command to change the 1st occurrence of "Hello there!" to "Goodbye" on the focus line and the next 9 lines.

`!/==!/`
: Locate a line-target of "==!".

`! nomsg all "!!READ THIS!!"`
: ALL command to display all lines containging string "!!READ THIS!!".

Command Reference Syntax Conventions

How to read the Syntax Diagrams

The following rules apply to the syntax diagrams used in this command reference.

1. The diagrams should be read from left to right, from top to bottom, following the path of the line.
   - The >>- symbol indicates the beginning of a statement.
   - The ->< symbol indicates the end of a statement.
2. Required items appear on the horizontal line (the main path).

```
>>--- required_item --------------------------------><
```

3. Optional items appear below the main path.

```
>>--- required_item ---+----------------------+-----><
                       |                      |
                       +-- optional_item -----+
```

4. If an optional item appears above the main path, then that item has no effect on the execution of the statement and is used only for readability.

```
                       +-- optional_item -----+
                       |                      |
>>--- required_item ---+----------------------+-----><
```

5. If you can choose from two or more items, they appear vertically, in a stack.
6. If you must choose one of the items, one item of the stack appears on the main path.

```
>>--- required_item ---+-- required_choice1 --+-----><
                       |                      |
                       +-- required_choice2 --+
```

7. If choosing one of the items is optional, the entire stack appears below the main path.

```
>>--- required_item ---+----------------------+-----><
                       |                      |
                       +-- optional_choice1 --+
                       |                      |
                       +-- optional_choice2 --+
```

8. If one of the items is the default, it appears above the main path and the remaining choices are shown below.

```
                       +-- default_choice ----+
                       |                      |
>>--- required_item ---+----------------------+-----><
                       |                      |
                       +-- optional_choice1 --+
                       |                      |
                       +-- optional_choice2 --+
```

9. An arrow returning to the left, above the main line, indicates an item that can be repeated.

```
                        +-----------------------+
                        v                       |
>>--- required_item ---+-- repeatable_item ---+----->< 
```

10. If the repeat arrow contains a comma, you must separate repeated items with a comma.

```
                        +- , -------------------+
                        v                       |
>>--- required_item ---+-- repeatable_item ---+----->< 
```

11. A repeat arrow above a stack indicates that you can repeat the items in the stack.

```
                        +-----------------------+
                        v                       |
>>--- required_item ---+-----------------------+-----><
                        |                       |
                        +-- optional_choice1 --+
                        |                       |
                        +-- optional_choice2 --+
```

12. Uppercase characters in keywords indicate the minimum abbreviation allowed for that particular command or parameter and must be spelled exactly as shown.

13. Variables appear in all lowercase letters and represent user-supplied names or values.

14. If punctuation marks, parentheses, arithmetic operators, or other such symbols are shown, you must enter them as part of the syntax.

15. Where a parameter immediately following a command verb begins with a non-alpha character, no separating blank is required between the command verb and the parameter. e.g. Add8, CHANGE/abc/xyz/

    Note: If CMDDEF=ALPHANUMERIC (default for INTERFACE=ISPF) is in effect, no separating blank is required only if the parameter immediately following a command verb begins with a non-alphanumeric character.

ADD

Syntax:

```
           +- 1 -+
           |     |
>>-- Add --+-----+---><
           |     |
           +- n -+
```

Description:

The ADD command adds one or more blank lines to the file. The lines are added after the focus line. The first line added becomes the focus line and the cursor is placed in column 1 of this line.

Parameters:

`n`	The number of lines to add. If omitted this parameter defaults to 1.

Example:

`a10`	Add 10 lines to the current file. **Note:** the minimum abbreviation (a) has been used for the command verb.

See Also:

SOS LINEADD
INPUT
Prefix command - A

ALL

Syntax:

```
>>-- ALL --+---------------+---><
           |               |
           +- line-target -+
```

Description:

The ALL command uses the selective line edit feature to allow the user to select only those lines which satisfy a condition defined by a target.

When a target is given, the selection level of all lines is first set to 0 and then the selection level of any line which satisfies the target is set to 1. Finally the SET DISPLAY 1 1 command is used to restrict the display to only those lines which have a selection level of 1. When used with no parameter, the selection level of all lines is set to 0 and the SET DISPLAY 0 0 is used so that all lines are selected.

Parameters:

`(null)`	Display all lines in the file.
`line-target`	Display only those lines which satisfy the line-target condition.

Example:

`all /x/|/y/`	Display only those lines which contain either an x or a y within the current zone limits.
`all changed`	Display only those lines which have been changed in the current edit session.

See Also:

MORE
LESS

If the target is a string target, the ALL command is affected by the following SET variable values:

ARBCHAR	Strings may contain wild card characters depending on this setting.
CASE	The case of the search strings will be respected or ignored depending on this setting.
HEXSTRING	Hex format search strings will be interpreted as such, if this is set on.
VARBLANK	If any of the search strings contain blanks then the search will be affected by this setting.
ZONE	Strings will be searched for only within the margins of the current zone.

ALLOCATE

Syntax:

```
>>-- ALLOCate --+---------+--- allocparms ---><
                |         |
                +- -Cat --+
                |         |
                +- -FREE -+
```

Description:

The ALLOCate command may be used to:

1. Dynamically define and/or allocate a data set.
2. Concatenate a list of data sets.
3. Concatenate a data set to an existing list of data sets.
4. Free (unallocate) a data set or override DISP/CLASS. (Same as FREE command.)

ALLOCATE allows users to allocate files whether or not a TSO environment is available. The syntax of the command closely matches that of the TSO ALLOCATE command so most ALLOCATE commands, executed without TSO as a prefix, will give the same results.

ALLOCATE is supported for MVS only.

Parameters:

`-CAT`	The data set being allocated is concatenated to an existing data set, or list of data sets, allocated to the specified ddname.
`-FREE`	Unallocates the data set(s) allocated to the specified ddname.
`allocparms`	Parameters supported by the TSO ALLOCATE command as follow:

```
DDname(ddname) or File(ddname), DAtaset('dsn'..) or DSName('dsn'..)
or DUMMY, MOD, NEW, OLD, SHR, CATALOG, DELETE, KEEP, UNCATALOG,
TRACKS, CYLinders, BLOCK, DIR(directory-blocks), SPACE(n[,m]),
VOL(volser[,volser...]), MAXVOL(volumes), UNIT(unit),
SYSOUT[(class)], WRITER(external-writer-name), FORMS(forms),
DEST(dest or node[.user]), COPIES(copies), BLKSIZE(blocksize),
LRECL(record-length), DSORG(PS|PO|DA), RECFM(format[,format...]),
BUFNO(buffers), OUTDES(output-descriptor-name),
STORCLAS(storage-class), MGMTCLAS(management-class),
DATACLAS(data-class), RECORG(LS), DSNTYPE(LIBRARY|PDS|HFS),
SPIN(UNALLOC) PATH(pathname), PATHOPTS(path-options-list),
PATHMODE(path-mode-list), PATHDISP(KEEP|DELETE[,KEEP|DELETE]),
FILEDATA(TEXT|BINARY), REUSE, LIKE('model-dsn')
```

In addition to these parameters, CBLe ALLOCATE supports the following:

`SUBSYS(subsys-name[,subsys-parm]...)`

Directs the allocation request to the specified subsystem name with optional parameters. Null parameters can be specified by leaving a subsys-parm empty.

Examples:

`alloc f(sysin) dsn('cbl.ssc.ctl(ssdemo01)') shr reuse`
Allocate an existing data set.

`alloc f(sysudump) da('nbj.sysudump') cyl space(100,20) lrecl(133) blksize(0) recfm(v,b,a) new catalog`
Allocate a new data set. Note that DCB information may be omitted.

`alloc f(multdsn) da('cbl.ssc.ctl(ssdemom1)' 'cbl.ssc.ctl(ssdemom2)') shr`
Allocate a new data set list.

`alloc -cat f(multdsn) dsn('cbl.cmx(nbj)')`
Allocate a new data set to the data set list allocated to ddname 'multdsn'.

`alloc -free f(multdsn)`
Unallocate data set(s) allocated to ddname 'multdsn'.

See Also:

FREE

BACKWARD

Syntax:

```
>>-- BAckward ----------------------><
```

Description:

The BACKWARD command scrolls the focus window backwards 1 page towards the top of the file.

Parameters:

None.

See Also:

FORWARD

BOTTOM

Syntax:

```
>>-- Bottom -----------------------------------------------------------><
```

Description:

BOTTOM is also an ISPF primary command.
If INTERFACE=ISPF is in effect (default for CBLi in an MVS environment), then the ISPF version of the command is executed instead. See section ISPF/CBLe CLI Command Precedence and IBM publication "ISPF: Edit and Edit Macros".
Use the CBLe CLI command ECOMMAND to override.

BOTTOM makes the last line in the file the focus line.

Parameters:

None.

See Also:

TOP

BROWSE

Syntax:

```
>>-- Browse -- fileid ---| SDE BROWSE Opts |--------------------------------><
```

Description:

For **VSE** and **CMS** systems, BROWSE is a synonym for VIEW.

Use the BROWSE command to open a Structured Data Environment (SDE) BROWSE window view to browse a page of data from the specified fileid.

Use BROWSE instead of VIEW to browse large data sets. Unlike EDIT and VIEW, BROWSE does not need to load the entire file into storage in order to display a page of records.

Parameters:

`fileid`
: The fileid of the file to be browsed.

  fileid may be the DSN of a sequential or VSAM data set, the member name (with or without the library DSN) of a PDS/PDSE member or an HFS file name. If member name is specified without a library DSN, the DSN of the library member in the current edit view is used.

`SDE BROWSE Opts`
: See SDE BROWSE for supported parameters.

CANCEL

Syntax:

```
>>-- CANCEL ----------------------------------------------------------------><
```

Description:

CANCEL is also an ISPF primary command.
If INTERFACE=ISPF is in effect (default for CBLi in an MVS environment), then the ISPF version of the command is executed instead. See section ISPF/CBLe CLI Command Precedence and IBM publication "ISPF: Edit and Edit Macros".
Use the CBLe CLI command ECOMMAND to override.

CANCEL causes CBLe to issue the QUIT command for all edited files in the edit ring.

If a file has been changed and not yet saved to disk (i.e. ALT is set to a value >0) then CBLe opens a dialog window asking whether you want to save the changes to the file before it is removed from memory.

Parameters:

None.

See Also:

QUIT

CAPPEND

Syntax:

```
>>-- CAppend --+----------+--><
               |          |
               +- string -+
```

Description:

Set the focus column to be the column immediately following the last character of the focus line and append the specified text to the focus line, starting at the the focus column.

Parameters:

`string`	Text string to be appended. Where more than 1 blank separates the text string from the command verb, the additional separating blanks are treated as part of the text string.

Example:

`ca abc`	Append "abc" to the focus line.
`ca  abc`	Append a single blank followed by "abc" to the focus line.
`ca abc def`	Append "abc def" to the focus line.

CBLI

Syntax:

```
>>-- CBLI --- command ---><
```

Description:

Execute a CBLi command.

The specified command is passed to the CBLi environment. Any windows opened are child windows of the CBLi main window not of the CBLe window.

Parameters:

`command`	CBLi command.

Example:

`cbli v cbl.vvc.ctl(vvrep01)`	Open a VCI window and execute CBLVCAT with control statements from file CBL.VVC.CTL(VVREP01).

See Also:

SYSCOMMAND

CDELETE

Syntax:

```
                +------- 1 -------+
                |                 |
>>-- CDelete --+-----------------+-->< 
                |                 |
                +- column-target -+
```

Description:

Delete characters from the current line starting at the focus column and continuing up to, but not including, the column-target.

Parameters:

`column-target`	Column-target condition.

Example:

`cd10`	Delete 10 columns from the focus line starting at the focus column. **Note:** the minimum abbreviation (cd) has been used for the command verb.

CFIRST

Syntax:

```
>>-- CFirst ---------------><
```

Description:

Position the focus column at the left zone column.

Parameters:

None.

Example:

`cf`	Set focus column to column 1. ZONE setting is: SET ZONE 1 *

See Also:

SET ZONE

CHANGE

Syntax:

```
>>-- Change -- /string1/string2/ ------------------------------------------->

 >-----+---------------------------------------------------------------------><
       |                                                  |
       +-- group-target --+-----------------------------------+
                          |                               |
                          +-- max_in_line --+-------------------+
                                            |                   |
                                            +-- first_in_line --+
```

Description:

CHANGE is also an ISPF primary command.
If INTERFACE=ISPF is in effect (default for CBLi in an MVS environment), then the ISPF version of the command is executed instead. See section ISPF/CBLe CLI Command Precedence and IBM publication "ISPF: Edit and Edit Macros".
Use the CBLe CLI command ECOMMAND to override.

Change occurrences of string1 to string2 on the focus line and on lines up to, but not including, the line containing the first match for group-target.

The "/" (slash) character is normally used as the delimiter encompassing and separating the string arguments. However, any non-alphanumeric character that does not have a special meaning to CBLe may be used as a string delimiter (e.g. "%", "#", "."). The delimiter character used must not appear in the either of string arguments.

Where the string arguments do not contain special or blank characters, blanks may be used as the delimiter character.

The length of the changed line may be increased or decreased where string1 < string2 and string1 > string2 respectively. Where the line length is increased, characters that extend beyond the truncation column are lost. By default, the truncation column is equal to the LRECL of the file.

The STAY setting determines which line is the focus line following a successful CHANGE command. If STAY is ON, the focus line remains unchanged. If STAY is OFF, the focus line is set to the last line of the target area (i.e. the line immediately preceding the group-target line.)

Where BLOCK is specified as the group-target, the CHANGE command will operate within the a currently-defined block. For line blocks, the CHANGE command operates on text within the current ZONE setting, whereas, for box blocks, the zone setting is ignored. Text is changed only if string1 is contained entirely within the block boundaries. Text outside the box will remain unchanged, i.e. it will not be shifted left or right due to unequal string argument lengths. Similarly, truncation may occur at the block's right column boundary if string1 < string2.

When issued from a macro, the CHANGE command sets the REXX stem variables CHANGE.0 to CHANGE.3.

`CHANGE.0`	3
`CHANGE.1`	Number of occurrences changed.
`CHANGE.2`	Number of lines changed.
`CHANGE.3`	Number of lines truncated. (Not yet supported.)

Parameters:

/string1/string2/
Source string and update string.
Note: settings for ARBCHAR, CASE, HEXSTRING and ZONE affect the search for string1.

/string1/ may be prefixed by one of the following special keywords:

`Word`	String1 must match a complete word.
`Prefix`	String1 must match the leading characters of a complete word.
`Suffix`	String1 must match the trailing characters of a complete word.

group-target
Group-target condition defining the end of the target area for the command. If the group-target is not satisfied then the CHANGE command will fail.
Default is 1.

max_in_line
Integer specifying the maximum number of occurrences of *string1* that may changed on an individual line in the target area. Alternatively, "*" (asterisk) may be specified to indicate that all occurrences of *string1* are to be changed.
Default is 1.

first_in_line
Integer specifying the first occurrence of *string1* in a line at which the change will be applied. Preceding occurrences of *string1* in the line remain unchanged.
Default is 1.

Example:

`change  Hello Hi`
Only change the 1st occurrence of the string "Hello" on the focus line to "Hi". The length of the line is reduced by 3 characters.

`change /abc/def/  ALL 1`
Change the 1st occurrence of the string "abc" on all lines.

`change /x'0D0A'/x'0000'/  ALL 1`
Where HEXSTRING is ON, the 1st occurrence of the 2 byte hex string x'0D0A' are changed to hex string x'0000' on all lines.

`change /X Y/z/  :100 2 3`
Change the 3rd and 4th occurrences of the string "X Y" on the focus line and all lines following, up to but not including line 100.

`change #A/B#X/Y#  ~/Ref:/ *`
Change all occurrences of the string "A/B" on the focus line and all lines following, up to but not including the first line that does not contain the string "Ref:"

`c word /All/Most/ 1 *`
Change all occurrences of the word "All" on the focus line. **Note:** "Allocate" will not be changed.

See Also:

SET ARBCHAR
SET CASE
SET HEXSTRING
SET ZONE

CINSERT

Syntax:

```
>>-- CInsert --+----------+--><
               |          |
               +- string -+
```

Description:

Insert a text string into the focus line starting at the focus column. Existing text in, or to the right of the focus column will be shifted to the right for the length of the inserted text string.

Parameters:

`string` Text string to be inserted.

Where more than 1 blank separates the text string from the command verb, the additional separating blanks are treated as part of the text string.

Example:

`ci abc`	Insert "abc" in the focus line at the focus column.
`ci  abc`	Insert a single blank followed by "abc" into the focus line.
`ci Hi Jane`	Insert "Hi Jane" in the focus line at the focus column.

CLAST

Syntax:

```
>>-- CLAst ---------------><
```

Description:

Position the focus column at the right zone column.

Parameters:

None.

Example:

`cla` Set focus column to column 50. ZONE setting is:

`SET ZONE 1 50`

See Also:

SET ZONE

CLIPBOARD

Syntax:

```
>>-- CLIPboard --+-------------------- CLEAR -------------------------+--><
                 |                                                     |
                 +-------------------- PASTE -------------------------+
                 |                                                     |
                 +-+------------+--+-- COPY ------------------------+-+
                   |            |  |                                 |
                   +-- APPEND --+  +-- CUT --------------------------+
                                   |                                 |
                                   +-- PUT --+-- BOX ---+-- string --+
                                             |          |
                                             +-- LINE --+
```

Description:

Move data to and from the CBLi environment clipboard.

The CLIPBOARD command is primarily for use in REXX edit macros and is also invoked from the **Edit** menu item in the CBLe Main Menu Bar.

Text can only be copied between clipboard and the file display area of a CBLe or SDE edit view. At this time, CBLi does not support moving text to and from a command line or non-CBLe/SDE edit view.

Before text can be copied to the clipboard, it must first be marked within the text display of an edit view. This may be done using prefix commands C(n), CC, M(n) or MM; or using <PF15> or <PF16> which are, by default, assigned to MARK BOX and MARK LINE respectively. An error message is displayed if there is no block marked in the current file.

If M(n) or MM is used to select a block of lines to be moved to the clipboard, then CLIPBOARD COPY is equivalent to CLIPBOARD CUT.

Parameters:

`APPEND`	Applicable to COPY, CUT and PUT to the clipboard only, append data to a new line following the existing clipboard data. If APPEND is not specified, existing clipboard data is replaced by the new text.
`CLEAR`	Empty the clipboard of all its contents.
`COPY`	Copy the marked block of text to the clipboard. The marked block of text is preserved in the file area.
`CUT`	Copy the marked block of text to the clipboard then delete the marked block of text from the file area.
`PASTE`	Insert the contents of the clipboard at the focus position within the file area. If the clipboard contains **any** text that has been inserted from a marked line block or from a CLIPBOARD PUT LINE operation, then the text is pasted into column 1 of new lines inserted following the focus line. If the clipboard contains only text inserted from a marked box block or CLIPBOARD PUT BOX operation, then the text is inserted at the focus column of the focus line and in as many lines that follow as is necessary to accomodate all lines in the clipboard. New lines are added immediately before the end of RANGE or end of file markers if necessary. Text in the file area to the right of the inserted text is shifted further to the right.
`PUT BOX|LINE` *string*	Copy the specified text string to the clipboard as a BOX block or as a LINE block. The effect of pasting a BOX block or LINE block is documented under PASTE. Where more than 1 blank separates the text string from the previous parameter, the additional separating blanks are treated as part of the text string.

Examples:

`clip copy`
Copy the currently marked block to the clipboard, replacing any existing clipboard text.

`clip append copy`
Copy the currently marked block to the clipboard, appending it to any existing clipboard text.

`clip append put box  This is some text.`
The string " This is some text." (1 preceding blank) is appended to a new line in the clipboard as a box block.

`clip clear`
Clear the contents of the clipboard.

See Also:

MARK

CLOCATE

Syntax:

```
>>-- CLocate --+-----------------+--><
               |                 |
               +- column-target -+
```

Description:

Locate and then position the focus column at a column target. In some cases, the focus line will also be set.

The focus line is scanned for column-target and, if successful sets the focus column to be the target column. If no match is found in the focus line and STREAM is set ON, then subsequent lines are scanned until a match is found, in which case the line containing the match becomes the focus line.

Where column-target is not specified, CBLe repeats the last CLOCATE command issued.

Parameters:

`column-target`	Column-target condition.

Example:

`cl 3`	Focus column is set 3 columns to the right of its current position.
`cl:50`	Focus column is set at column 50.
`cl/Hello/`	Focus column is set at the start of the first occurrence of the string "Hello".

CMSG

Syntax:

```
>>-- CMSG --+------------+-><
            |            |
            +-- string --+
```

Description:

Place a text string on the command line. CMSG is most often used in macros to place a command on the command line.

If no text string is specified, then the command line is cleared.

Parameters:

`string`	Text string to be placed on command line.

Example:

`cmsg 'ALL /test data/'`	Place the string "ALL /test data/" on the command line.

COMMAND

Syntax:

```
>>-- COMMAND -- command ------->< 
```

Description:

The COMMAND command temporarily disables CBLe's synonym processing.

When SYNONYM ON is in effect, a command entered from a CBLe command line is automatically checked to determine whether it is the name of a defined synonym. If so, the action taken will be that specified by the synonym definition.

Prefixing a command or macro name with COMMAND will bypass synonym checking for the command/macro being executed.

Parameters:

`command`	Any CBLe command or macro name followed by its parameters.

Example:

`command ALL /=/`
The CBLe command ALL is executed without checking whether a synonym for ALL exists.

See Also:

SYNEX
SET SYNONYM

COMPARE

Syntax:

```
>>-- COMPare -- fileid1 --- fileid2 ------><
```

Description:

Compare lines of text in two files that are displayed in existing edit views within the current CBLe edit environment.

Text starting at the left ZONE boundary of the line following the current line in file 1 is compared with text starting at the left ZONE boundary of the line following the current line in file 2.

Where the zone widths of the two files differ, then, for the compare operation only, the lines belonging to the file with the shorter zone are assumed to be padded with blanks. The text compare is case sensitive unless SET CASE MIXED|UPPER IGNORE is in effect in **both** file edit views.

If the compared lines match, then the text in the next line in sequence following the current line on file 1 is compared with the equivalent line in file 2. This process is repeated until either the lines being compared do not match or the end of RANGE is encountered for one of the files.

Note that, where the RANGE is not explicitly set for an edited file, then the "End of File" line is the end of range. If end of range is encountered on one file before it is encountered on the other, then a difference is flagged.

If no differences are found, the current line is unchanged, otherwise the first line that contains a difference becomes the current line in both views.

Parameters:

`fileid1`	The fileid of the first file to be compared. The file must exist in an edit view.
`fileid2`	The fileid of the second file to be compared. The file must exist in an edit view.

Example:

`compare   CBL.ACCT.X017993.T070125   CBL.ACCTLIB(X017993)`
Compare text in a sequential file with that in a PDS member.

See Also:

SET RANGE
SET ZONE

COPY

Syntax:

```
>>-- COpy -- group-target --+---------------+--><
                            |               |
                            +- line-target -+
```

Description:

COPY is also an ISPF primary command.
If INTERFACE=ISPF is in effect (default for CBLi in an MVS environment), then the ISPF version of the command is executed instead. See section ISPF/CBLe CLI Command Precedence and IBM publication "ISPF: Edit and Edit Macros".
Use the CBLe CLI command ECOMMAND to override.

Copy text from a target area to a target line.

Where BLOCK is specified, COPY supports copying text between files. Otherwise COPY can only operate on lines in the same edited file. The first line of the copied block becomes the focus line.

Where BLOCK is not specified as the group-target, the last line of the copied target group of lines becomes the focus line.

Where line-target is omitted, the current focus line is used.

Parameters:

`group-target`

Group-target condition defining the end of the source target area. If the group-target is not satisfied then the COPY command will fail.
If BLOCK is specified, then the marked block will be copied as follows:

Line Block
Marked line(s) are copied to the line immediately following the line-target.

Box Block
Marked box is copied to the focus column of the target line.

Note: group-target key word ALL is not supported.

`line-target`

Line-target condition defining a destination target line. If the line-target is not satisfied then the COPY command will fail.
Where the source target area is not a box block, lines are copied to the line immediately following the target line.

Example:

`copy 8 :28`	The focus line and the 7 lines following are copied to line below line 29.
`co -4 6`	The focus line and the 3 lines preceding are copied below the 5th line following the focus line.
`co 3 -/SELC/`	The focus line and the 2 lines following are copied to the line following the first line containing the string "SELC", scanning backwards from the focus line.
`co prefix /call/ word /state/`	The focus line and all lines up to, but not including, the first line to contain a word beginning "call" are copied to the line below the first line following the focus line that contains the word "state".
`co block`	The marked box block is copied to the focus column of the focus line and lines that follow up to the depth of the marked block.

See Also:

MOVE

COUNT

Syntax:

```
                                 +------- 1 ------+
                                 |                |
   >>-- COUnt -- /string/ ----+----------------+-->< 
                                 |                |
                                 +- group-target -+
```

Description:

Count occurrences of string on the focus line and on lines up to, but not including, the line containing the first match for group-target.

The "/" (slash) character is normally used as the delimiter encompassing and separating the string arguments. However, any non-alphanumeric character that does not have a special meaning to CBLe may be used as a string delimiter (e.g. "%", "#", "."). The delimiter character used must not appear in the either of string arguments.

Where the string arguments do not contain special or blank characters, blanks may be used as the delimiter character.

The STAY setting determines which line is the focus line following a successful COUNT command. If STAY is ON, the focus line remains unchanged. If STAY is OFF, the focus line is set to the last line of the target area (i.e. the line immediately preceding the group-target line.)

Where BLOCK is specified as the group-target, the COUNT command will operate within the a currently-defined block. For line blocks, the COUNT command operates on text within the current ZONE setting, whereas, for box blocks, the zone setting is ignored. The count is incremented only if string is contained entirely within the block boundaries.

When issued from a macro, the COUNT command sets the REXX stem variables COUNT.0 to COUNT.2.

`COUNT.0`	2
`COUNT.1`	Number of occurrences of string.
`COUNT.2`	Number of lines containing at least one occurrence of string.

Parameters:

`/string/` The search string. **Note:** settings for ARBCHAR, CASE, HEX and ZONE affect the search for string.

/string/ may be prefixed by one of the following special keywords:

Word String must match a complete word.

Prefix String must match the leading characters of a complete word.

Suffix String must match the trailing characters of a complete word.

`group-target` Group-target condition defining the end of the target area for the command. If the group-target is not satisfied then the COUNT command will fail.

Example:

`count esc`	Count occurrences of the string "esc" on the focus line.
`count /abc/ ALL`	Count occurrences of the string "abc" on all lines.
`count X'C1' :100`	Count occurrences of the hex string X'C1' (Character 'A') on the focus line and all lines following, up to but not including line 100.
`count #A/B# ~/Ref:/`	Count occurrences of the string "A/B" on the focus line and all lines following, up to but not including the first line that does not contain the string "Ref:"
`cou prefix /re/ all`	Count occurrences of the words beginning "re" on all lines. **Note:** "irrelevant" will not be counted.

See Also:

SET ARBCHAR
SET HEXSTRING
SET CASE
SET ZONE

COVERLAY

Syntax:

```
>>-- COVerlay --- string --><
```

Description:

Overlay text in the focus line with the specified text string starting at the focus column.

Characters in the focus line that correspond to blanks in the supplied overlay string are unchanged. In order to overlay a character with a blank, the "_" (underscore) character should be specified in the corresponding position of the overlay string.

All other characters supplied in the overlay string replace characters in corresponding positions in the focus line.

Parameters:

`string`	Overlay text string. Where more than 1 blank separates the text string from the command verb, the additional separating blanks are treated as part of the text string.

Example:

`cov abc`	Overlay text at focus line and column with "abc".
`cov  ABC_X YZ`	Original text at focus line and column is: 0123456789 After column overlay command text is: 0ABC X6YZ9

CREATE

Syntax:

```
>>-- CREate -- BUTton --+----------+--+---------------------+-- /name/command/ --><
                        ^          |  |                     |
                        |          |  | +-- 0 --+ +-- 1 --+ |
                        +- (OptA) -+  | |       | |       | |
                                      +-+- row -+-+-------+-+
                                                  |       |
                                                  +- col -+
```

OptA

```
  |--+--+- COLour -+---- (OptB) ----+--->
     |  +- COLor --+                |
     |                              |
     +---- PRESsed ----- (OptB) ----+
     |                              |
     +---- CURsor ------------------+
     |                              |
     |  +- KEEP ---+                |
     |  |          |                |
     +--+----------+----------------+
        |          |
        +- PASS ---+
```

OptB

```
     +- Blue (1) --+
     +- White (2) -+  +- REVvideo --+
     |             |  |             |
  |--+- Default ---+--+-------------+--->
     |             |  |             |
     +- Green -----+  +- Blink -----+
     |             |  |             |
     +- Pink ------+  +- NONe ------+
     |             |  |             |
     +- Red -------+  +- Uscore ----+
     |             |
     +- Turquoise -+
     |             |
     +- Yellow ----+

   (1) Default for PRESSED.
   (2) Default for COLOUR.
```

Description:

CREATE is also an ISPF primary command.
If INTERFACE=ISPF is in effect (default for CBLi in an MVS environment), then the ISPF version of the command is executed instead. See section ISPF/CBLe CLI Command Precedence and IBM publication "ISPF: Edit and Edit Macros".
Use the CBLe CLI command ECOMMAND to override.

Create a selectable button within the MDI parent display which, when pressed, executes the associated command or macro.

The "/" (slash) character is normally used as the delimiter encompassing and button name and associated command. However, any non-alphanumeric character that does not have a special meaning to CBLe may be used as a string delimiter (e.g. "%", "#", "."). The delimiter character used must not appear in the name or command strings.

CBL distributes the edit macro PROFIRST which uses the CREATE BUTTON command to add some useful buttons to the CBLe and SELCOPY Interactive MDI windows when they are first opened.

Parameters:

`COLOUR`
`COLOR`
Define the colour and extended highlighting to be applied to the button text when it is in an unselected state.

Default is WHITE REVVIDEO.

`PRESSED`
Define the colour and extended highlighting to be applied to the button text when it is in a selected (pressed) state.

The button is in a selected state for the duration of the command execution after which the button reverts back to being in the unselected state. Unless the screen is refreshed during execution of the command (e.g. if CURSOR specified), the user does not see the button in a selected state.

Default is BLUE REVVIDEO.

`CURSOR`
Delay execution of the command until the <Enter> key is hit, allowing the user to position the cursor in the display area within an edit view prior to executing the command. CURSOR should be used where the command is sensitive to the cursor position.

If the cursor is positioned on a command line when <Enter> is hit or if any other AID key is hit while the cursor is within an edit view, then the button reverts back to the unselected state and the command is discarded.

`KEEP`
`PASS`
Prior to executing the command, either keep the cursor in its current position when the button is selected (i.e. positioned on the button) or PASS it back to the command line of the current edit view.

Default is KEEP.

`row`
Row within the MDI parent display at which the button is to be positioned. Rows are numbered from 0 (zero) to the depth of the MDI parent display area where row 0 is the row containing the menu bar.

Default is 0.

`col`
Column within the MDI parent display at which the button is to be positioned. Columns are numbered from 0 (zero) to the width of the MDI parent display area where column 1 is the column containing the system menu button.

Default is 1.

`name`
Defines the name assigned to the button and the text displayed on the button itself.

`command`
The command string to be executed. This may be any command string supported by the current edit view.

Where more than one command is to be executed, they should be invoked via a macro name (in storage or saved to disk) or via the IMMEDIATE command. The LINEND character is not respected in a CREATE BUTTON command.

Example:

```
create button 0 80 /HW/msg Hello World/
crea but col yell uscore press yell rev 1 1 /HD/macro HD/
crea but col gr non 1 5 /Ring/imm 'ext ring';do i=1 to ring.0;'msg' ring.i;end/
crea but col bl non press wh rev cursor 1 10 /ColN/imm 'ext cursor';'msg' cursor.2/
crea but press turq rev pass 0 85 #CL#cmsg imm 'stream on';'cl/xxx/';'stream off'#
```

See Also:

DESTROY

CREPLACE

Syntax:

```
>>-- CReplace --- string --><
```

Description:

Replace text in the focus line with the specified text string starting at the focus column.

All characters supplied in the text string, incuding leading, imbedded and trailing blanks, replace characters in corresponding positions in the focus line.

Parameters:

`string`	Replace existing text with this text string. Where more than 1 blank separates the text string from the command verb, the additional separating blanks are treated as part of the text string.

Example:

`cr abc`	Replace text at focus line and column with "abc".
`cr  ABC_X YZ`	Original text at focus line and column is: 0123456789 After column replace command text is: ABC_X YZ9

CURSOR

Syntax:

```
>>-- CURsor --+- HOME ---------------+--><
              |                      |
              +- Column -------------+
              |                      |
              +- CMdline ----+-----+-+
              |              |     | |
              |              +- n -+ |
              |                      |
              +- File -- m --+-----+-+
                             |     |
                             +- n -+
```

Description:

Position the cursor within the current edit window view. The CURSOR command is most often used in macros.

Note: The CURSOR command does not alter the window view itself. i.e. the current line and column remains unchanged.

Parameters:

`HOME`	If the cursor is on the command line, the cursor is positioned at the line and column number at which it was positioned when it was last in the file area. If this line is not in the current edit view, the cursor is positioned in column 1 of the current line.

If the cursor is in the file area, the cursor is positioned at column 1 of the command line.

`COLUMN` The cursor is positioned at the focus column of the focus line.

`CMDLINE [n]` The cursor is positioned at column 1 of the command line. If a column number "n" is specified, the cursor is positioned at this column number of the command line.

`FILE m [n]` The cursor is positioned at column 1 of line number "m" of the file. If a column number "n" is specified, the cursor is positioned at this column number of line number "m".

If the specified file line or column number is outside the current file display area, then error message **EDT044E** is returned. e.g.

```
EDT044E Invalid cursor line or column in command cursor file 138 20.
```

DEFINE

Syntax:

```
>>-- DEFine --+----------------------------+-->< 
              |                            |
              +- fileid -------------------+
              |                            |
              +- macroname -+--------------+
                            |              |
                            +-- macrodef --+
```

Description:

DEFINE is used to load an existing CBLe REXX macro from disk into storage or to assign a temporary macro definition in storage.

DEFINE, with no parameters, will display the macro names of macros that have been loaded into storage.

Keeping a copy of a macro in storage is beneficial when it is to be called frequently. The disk I/O involved in loading the macro is not repeated each time the macro is called and so performance is improved. However, keeping a macro in storage will incur storage overheads.

Use the PURGE command to remove macros from storage.

Parameters:

`fileid` The full fileid of the macro to be loaded into storage.

`macroname` The macroname (filename) of the macro file on disk, or the name assigned to a new macro definition, to be loaded into storage.

The macroname must adhere to local environment naming specifications.

`macrodef` The macro definition to be assigned to macroname. A macrodef is a short CBLe REXX macro that may be defined on a single line.

Where macrodef is omitted, CBLe searches the default macro libraries for a file name that matches the macroname supplied and having file type CBLE. The default macro libraries are determined as follow:

- On MVS, the parameter MacroPath in the Edit section of CBLIINI.
- On VSE, the library path specified by the LIBDEF PROC search chain.
- On CMS, all accessed mindisks.

Example:

```
define  prd2.cbl210.puttime.cble
```

Load "puttime" macro into storage from VSE sublib "prd2.cbl210".

```
def delsame
```

Scan macro libraries for macro "delsame" and load into storage.

```
define
```

Display list of loaded macros.

```
define allnext 'wrap off'; ':1';  do forever; 'nomsg/##/'; if rc0 then leave; '1'; 'sel 5'; end; 'disp 5'
```

Define a temporary macro to display all lines that immediately follow lines containing "##", assign it the name "allnext" and load it into storage.

See Also:

PURGE
IMMEDIATE

DELETE

Syntax:

```
              +------- 1 ------+
              |                |
>>-- DELete --+----------------+-->< 
              |                |
              +- group-target -+
```

Description:

DELETE is also an ISPF primary command.
If INTERFACE=ISPF is in effect (default for CBLi in an MVS environment), then the ISPF version of the command is executed instead. See section ISPF/CBLe CLI Command Precedence and IBM publication "ISPF: Edit and Edit Macros".
Use the CBLe CLI command ECOMMAND to override.

Delete one or more lines from the edited file starting at the focus line.

Parameters:

`group-target`	Group-target condition defining the end of the source target area. If the group-target is not satisfied then the DELETE command will fail.

Example:

`delete`	Delete the focus line only.
`del :10`	Delete the focus line and all lines up to, but not including, line 10.
`del /abc/`	Delete the focus line and all lines up to, but not including, the first line to contain the string "abc".

DESTROY

Syntax:

```
>>-- DEStroy -- BUTton --+--/name/--+--><
                         |          |
                         +--- * ----+
```

Description:

Destroy a specified button or all buttons generated by a previous CREATE BUTTON command.

As for CREATE BUTTON, any supported delimiter character may be used to encompass the name string so long as the delimiter character used does not appear in the name string.

Parameters:

`name`	The name of the button to be destroyed.
`*`	(asterisk) indicates that all buttons are to be destroyed.

Example:

```
destroy button /HW/
destroy button *
```

See Also:

CREATE

DIALOG

Syntax:

```
>>-- DIALOG -- /prompt/ --+-------------------------------------+-+----------------+-->
                          |                                     | |                |
                          +- EDITfield -+----------+-+------------+ +- TITLE /title/-+
                                        |          | |            |
                                        +- /text/ -+ +- PASSWORD -+
```

```
     +------ OK ------+  +- DEFButton 1 -+
     |                |  |               |
 >---+----------------+--+---------------+--+-------------------+----><
     |                |  |               |  |                   |
     +--- OKCANCEL ---+  +- DEFButton n -+  +- ICONExclamation -+
     |                |                     |                   |
     +---- YESNO -----+                     +- ICONInformation -+
     |                |                     |                   |
     +- YESNOCANCEL --+                     +- ICONQuestion ----+
                                            |                   |
                                            +- ICONStop --------+
```

Description:

For use in CBLe macros only, DIALOG opens a dialog window prompting the user to enter text and/or select an action.

The "/" (slash) character is normally used as the delimiter encompassing the charecter string for **prompt**, **text** and **title**. However, any non-alphanumeric character that does not have a special meaning to CBLe may be used as a string delimiter (e.g. "%", "#", "."). The delimiter character used must not appear in the string argument.

The following REXX compound variables, which do not require an EXTRACT command, are set on completion of the DIALOG command:

`dialog.0` 2

`dialog.1` The contents of the edit field when the dialog window is closed. If no edit string is entered or if no edit field is present, then this variable is set to the null string.

`dialog.2` The upper case text string of the action selected when the dialog window is closed. i.e. "OK", "CANCEL", "YES", "NO" or "PF3". ("PF3" indicates that the dialog window was closed by the user without selecting an action.)

Parameters:

`prompt`	Character string, within delimiters, to be displayed in the dialog window prompting the user for a response. The delimiters are not included as part of the character string.
`EDITFIELD`	Display an editable field in the dialog window allowing the use to type a response string.
`text`	Character string, within delimiters, to be displayed in the editable field when the dialog window is opened. The delimiters are not included as part of the character string.
`PASSWORD`	Hide the character string typed in the edit field so as not to display potentially sensitive data.
`TITLE /title/`	Display a title at the top of the dialog window. Character string, within delimiters, defining text to be displayed in the title field. The delimiters are not included as part of the character string.

`OK`
`OKCANCEL`
`YESNO`
`YESNOCANCEL`
	Defines the action buttons to be displayed at the bottom of the dialog window. OK is the default unless EDITFIELD is specified in which case OKCANCEL is default.

`DEFBUTTON n`	Default button number. The action buttons at the bottom of the dialog window are numbered from left to right. DEFBUTTON defines which of these buttons are default. Where EDITFIELD is not specified, the cursor is positioned on the default button. Default is button number 1.

`ICONExclamation` `ICONInformation` `ICONQuestion` `ICONStop`	Display one of the standard CBLe icons in the dialog window. These are "!" (exclamation mark), "i", "?" (question mark) or "STOP".

Example:

```
'dialog  /Enter Your Password./  editfield  password   title /Protected File/  OKCANCEL  iconq'
 if dialog.2  = 'CANCEL'    |   dialog.1  'open sesame'   then
   do;   call AuthFailed;   exit;   end
```

```
 ___________________________________________
| Protected File                          x |
|                                           |
|  ?  Enter Your Password.                  |
|                                           |
|     > _________________________________   |
|                                           |
|           OK             Cancel           |
|                                           |
'___________________________________________'
```

See Also:

POPUP

DOWN

Syntax:

```
                      +-- 1 --+
                      |       |
>>--+-- Down --+----+-------+-->< 
    |          |    |       |
    +-- Next --+    +-- n --+
```

Description:

DOWN is also an ISPF primary command.
If INTERFACE=ISPF is in effect (default for CBLi in an MVS environment), then the ISPF version of the command is executed instead. See section ISPF/CBLe CLI Command Precedence and IBM publication "ISPF: Edit and Edit Macros".
Use the CBLe CLI command ECOMMAND to override.

DOWN positions the focus line one or more lines below the current focus line.

Where the specified number is greater than the number of lines below the focus line, the End of File line becomes the focus line.

Parameters:

`n`	The number of lines below the current focus line at which the new focus line is to be set. If omitted this parameter defaults to 1.

Example:

`down`	Set the focus line to be 1 line below the current focus line.
`d10`	Set the focus line to be 10 lines below the current focus line.

See Also:

UP

DUPLICATE

Syntax:

```
                   +-- 1 --+  +-------- 1 ------+
                   |       |  |                 |
>>-- DUPlicate --+-------+--+-----------------+--><
                   |       |  |                 |
                   +-- n --+  +- group-target -+
```

Description:

Duplicate lines in the file one or more times. Lines to be duplicated are defined by group-target.

Duplicated lines are inserted immediately following the defined group-target area. The first duplicated line becomes the focus line.

Parameters:

`n`	The number of times the line or group of lines are to be duplicated. If omitted this parameter defaults to 1.
`group-target`	Group-target condition defining group of lines to be duplicated. If the group-target is not satisfied then the DUPLICATE command will fail. BLOCK is supported as a group-target for DUPLICATE for line blocks only.

Example:

`duplicate`	Duplicate the focus line once.
`dup 5`	Duplicate the focus line 5 times.
`dup 3 /SELC/`	Duplicate 3 times the focus line and lines up to, but not including, the first line to contain the string "SELC".
`dup 1 BLOCK`	Duplicate the marked line block once.

ECOMMAND

Syntax:

```
>>-- ECommand -- command ------><
```

Description:

Where a command exists in both the CBLE and ISPF edit command sets and INTERFACE=ISPF is active, ECOMMAND may be used as a prefix in order to force the use of the CBLe version of the command.

If INTERFACE=CBLE is active, the ECOMMAND prefix has no effect.

Parameters:

`command`	Any CBLe command followed by its parameters.

Example:

`ECommand Change /ABC/DEF/ * *`
The CBLe version of the CHANGE command is used regardless of INTERFACE=ISPF.

See Also:

ICOMMAND

EDIT

Syntax:

```
>>--+- Edit --+--+---------------------------------------------------------+--><
    |         |  |                                                         |
    +- Kedit -+  +-------------------    - (minus) -------------------------+
    |         |  |                                                         |
    +- Xedit -+  +- fileid -+---------------------------------------------+
                            |                                             |
                            +- ( +----------------------+-+----------------+
                                 |                      | |                |
                                 +- PROFILE macroname -+ +- | HFS Opts | -+
                                 |                      |
                                 +- NOPROFile ---------+
```

HFS Opts:

```
        +- EOL ---+------------+---------------+
        |         |            |               |
        |         +-- CR ------+               |
        |         |            |               |
        |         +-- LF ------+               |
        |         |            |               |
        |         +-- NL ------+               |
        |         |            |               |
        |         +-- CRLF ----+               |
        |         |            |               |
        |         +-- LFCR ----+               |
        |         |            |               |
        |         +-- CRNL ----+               |
        |         |            |               |
        |         +-- string --+               |  +- LRECL lrecl -+
        |                                      |  |               |
 >------+--------------------------------------+--+---------------+----->
        |                                      |
        +- RECFM ---- F -----------------------+
```

Description:

X is also a synonym for the ISPF primary command EXCLUDE.
If INTERFACE=ISPF is in effect (default for CBLi in an MVS environment), then the ISPF EXCLUDE is executed instead. See section ISPF/CBLe CLI Command Precedence and IBM publication "ISPF: Edit and Edit Macros".
Use the CBLe CLI command ECOMMAND to override.

Edit a specific file or place focus on the next or previous CBLe edit view in the edit ring. The CBLe edit view that has the focus following execution of EDIT, is the new CBLe current window.

If *fileid* is the same as that of a file which is already in the ring of edited files, then focus is placed on the edit view containing that file. Otherwise, a file with this fileid is read from disk and a new edit view is created to edit the data. If no matching file is found on disk, then a new, empty edit view is opened for this fileid.

Where a single token is specified for *fileid*, CBLe treats it as the FNAME portion of a fileid. The file edited will have a fileid with the specified FNAME with all other portions of the fileid (FPATH, FTYPE and FMODE) being equal to that of the file in the current edit view.

If no parameters are specified, the next CBLe edit view in the edit ring becomes the current CBLe edit view. If "-" (minus) is specified, the previous CBLe edit view in the edit ring becomes the current CBLe edit view. Note that CBLe edit views are just one of the child window types supported by the CBLe MDI parents. Other types, not included in the CBLe edit ring, include SDE window views, HELP windows and List windows.

Where *fileid* is identified as being an HFS path (i.e. begins with "." or contains "/"), HFS options may be specified to determine the handling of data within the file.

For successful edit, the user must have read/write access to the requested file and the file file must not be locked (exclusively enqueued). If the file to be edited is already being used exclusively by another process or user, then a dialog window is opened which offers the user read-only edit (i.e. VIEW) of the file data instead.

The maximum number of files that may be edited is restricted by system resources only.

Parameters:

fileid

The fileid of the file to be edited and made the current file in the ring of edited files. Fileid may be any of the file formats supported by CBLe on the local system (as described by SET FILEID.)

PROFILE *macroname*

The REXX edit macro to be executed as the profile when editing the file.

The macro name must exist in a library within the CBLe macro path.

The PROFILE option only affects the profile for the file that is being added to the edit ring, and does not affect the profile to be used when additional files are added to the edit ring later in the edit session.

macroname overrides use of the default profile macro defined by the CBLiINI Edit.DefProfile and/or the CBLe command SET DEFPROFILE (default PROFILE.)
Default is macro name, PROFILE.

NOPROFILE

Suppress the use of a profile macro when editing the file.

The NOPROFILE option only affects the file currently being added to the ring, and does not affect the profile to be used when additional files are added to the ring later in the edit session.

`EOL=NL|CR|LF|CRLF|LFCR|CRNL|`*`string`*

Sets the EOLIN (input end-of-line) delimiter value used to determine the end of each record for non-RECFM F input. EOLIN delimiters are not included in the edited record data or record length. EOL parameter elements are as follow:

NL	X'15'	New Line.
CR	X'0D'	Carriage Return.
LF	X'0A'	Line Feed.
string	-	A 2-byte user specified character or hex string.

Default is the current value for EOLIN.

`RECFM F`

Specifies that the data is to be treated as containing Fixed length format records as defined by the LRECL argument.

`LRECL `*`lrecl`*

Specifies the maximum record length of input records.

Records terminated by an EOL sequence will wrap onto the next line of data if the record length exceeds *lrecl*. Where a record has wrapped, the prefix area contains the "==EOL>" flag. Furthermore, read-only edit is forced in order to suppress save of a wrapped record as multiple, individual records.

For RECFM F data, *lrecl* is the fixed length of the records in the edit view. If the file size is not a multiple of the lrecl value an error occurs and edit is cancelled.
Note: Use VIEW to display data as fixed format with the last record padded with blanks up to the lrecl length.

For EOL delimited records, default lrecl is 32752. Otherwise, for RECFM F, default lrecl is 80.

Example:

`edit`

Make the next file in the ring the current file.

`edit-`

Make the previous file in the ring the current file.

`edit cbl.cmd(temp)`

Edit member "TEMP" of MVS PDS "CBL.CMD".

`edit cbl.jcl(ssfind)  (profile jclprof`

Edit member "SSFIND" of MVS PDS "CBL.JCL" and override the default profile macro with macro "JCLPROF".

See Also:

SET FILEID

EDITV

See also the **EQU** CBLe REXX macro, found in the CBL distributed macro library, which utilises the EDITV command. This macro provides the user with an intuitive method of setting, verifying and unsetting CBLe user environment variables at the global level. Execute **em equ** from a CBLe edit view for EQU arguments.

Syntax:

```
                              +-----------------------+
                              v                       |
>>-- EDITV ---+--+-- GET ---+------ varname -----------+-+----><
              |  |          |                            |
              |  +-- GETF --+                            |
              |  |          |                            |
              |  +-- PUT ---+                            |
              |  |          |                            |
              |  +-- PUTF --+                            |
              |               +-----------------------+ |
              |               v                       | |
              +--+-- SET ---+------ varname -- value --+-+
              |  |          |                            |
              |  +-- SETF --+                            |
              |                                          |
              +--+-- SETL --+------ varname -- value ----+
              |  |          |                            |
              |  +-- SETLF -+                            |
              |  |          |                            |
              |  +-- SETFL -+                            |
              |                 +-----------+            |
              |                 v           |            |
              +--+-- LIST --+----+-----------+-----------+
                 |          |    |           |
                 +-- LISTF -+    +- varname -+
```

Description:

Set, retrieve and list CBLe user environment variables at both a file and global level.

EDITV may be used to share information between edit macros executed at different times in the same CBLe edit session. Variables set using EDITV may also be referenced for substitution in any command issued from an edit view command prompt or from within edited data via the CMDTEXT facility. e.g.

```
editv  setl worklib  sys8.temp.userdata
listmembers  %worklib%
```

File edit variables apply only to the individual files in which they were set. Groups of file variables are maintained for each file in the ring and each file's variables are kept until the file is removed from the edit ring.

Global edit variables apply to all files in the edit ring and are kept until the CBLe application is closed.

Parameters:

`SET`
`SETF`
SET/SETF is used in an edit macro or executed from a command line to set a variable name to a value which must be a string consisting of a single token enclosed in blanks.

Where a variable is specified without a value, the variable will be unset (set to null).

SET assigns edit variables at the global level.
SETF assigns edit variables at the file level.

`SETL`
`SETLF`
`SETFL`
SETL/SETLF(SETFL) performs the same function as SET/SETF except that only one variable may be set for each EDITV SETL/SETLF command and the value may be a string containing any number of tokens.

SETL assigns an edit variable at the global level.
SETLF (synonym SETLF) assigns an edit variable at the file level.

`PUT`
`PUTF`
PUT/PUTF may be used in an edit macro only, to assign the values of one or more REXX macro variables to edit variables of the same name. e.g.

REXX macro variables are set as follow:
```
HOST = Saturn ; A = 12+8
```
The same variable names and values can be set to edit variables using 'EDITV PUT HOST A'.

PUT assigns edit variables at the global level.
PUTF assigns edit variables at the file level.

`GET`
`GETF`
GET/GETF may be used in an edit macro only, to assign the values of one or more edit variables to REXX macro variables of the same name.

GET retrieves global edit variables.
GETF retrieves file edit variables.

`LIST` `LISTF`	LIST/LISTF is used to list the specified edit variable names and their values to the CBLe message lines. Default is all applicable variable names. LIST will list global edit variables. LISTF will list file edit variables.
`varname`	Edit variable name.
`value`	Edit variable string value. Must be a single token for SET/SETF.

Examples:

```
editv set tmp nbj.tmp work nbj.work.txt user guest

editv setl docs All of this is assigned to "docs"

editv list
```

EMSG

Syntax:

```
    >>-- EMSG --+------------+---->< 
                |            |
                +-- string --+
```

Description:

Display a message string on the message line as an error message. EMSG is most often used in CBLe macros.

If BEEP is ON then EMSG will trigger the 3270 terminal alarm.

If no text string is specified, then the message line is cleared.

Parameters:

`string`	Text string to be displayed on the message line.

Example:

emsg 'Error.. Timestamp mismatch'

See Also:

MSG

EXTRACT

Syntax:

```
               +------------+
               |            |
               v            |
>>-- EXTract --+-- option --+-------------------------------------------------><
```

Description:

Extract information about the current edit environment for use in CBLe macros.

The values for each EXTRACT option are assigned to REXX stem variables.

Parameters:

option
Each EXTRACT option must be separated with a delimiter which may be any non-alphanumeric character, including blank.

Currently supported EXTRACT options and their returned REXX variables are as follow:

Alt	alt.0	2
	alt.1	Number of alterations since last AUTOSAVE or SAVE.
	alt.2	Number of alterations since last SAVE.
ARBchar	arbchar.0	3
	arbchar.1	ON|OFF
	arbchar.2	First ARBCHAR character.
	arbchar.3	Second ARBCHAR character.
BEEP	beep.0	1
	beep.1	ON|OFF
BLOCK	block.0	8
	block.1	LINE|BOX|NONE
	block.2	Line number of start of block. (Null if block.1=NONE).
	block.3	Column number of start of block. (Null if block.1=NONE).
	block.4	Line number of end of block. (Null if block.1=NONE).
	block.5	Column number of end of block. (Null if block.1=NONE).
	block.6	Fileid of file containing marked block. (Null if block.1=NONE).
	block.7	PERSISTENT (Null if block.1=NONE).
	block.8	Contents of a block that spans 1 line only. (Null if block.1=NONE or marked block spans multiple lines).
CASE	case.0	3
	case.1	MIXED|UPPER
	case.2	RESPECT|IGNORE
	case.3	RESPECT|IGNORE
CMDDEF	cmddef.0	1
	cmddef.1	ALPHA|ALPHANUMERIC
CMDText	cmdtext.0	3
	cmdtext.1	The text of the command that would be either executed or placed on the command line had the CMDTEXT command been issued with respect to the current focus column and line.
	cmdtext.2	'<' indicating that the command should be executed immediately, or '>' indicating that the command should be placed on the command-line.
	cmdtext.3	The placement position of the cursor within the resulting command had it been put on the command line, as defined by the presence of the 1st underscore character (_) in the original source field. Note that the first underscore is removed from the resulting command. If no underscore is present then cmdtext.3 will be 0.
COLOr	color.0	Number of items reflecting the SET COLOUR commands in effect for the current file.
	color.i	Each SET COLOUR item, field name in uppercase, followed by its colour and extended highlighting code. See SET COLOUR.
COLOur	colour.n	Same as COLOR.
COLumn	column.0	1
	column.1	Column number of the focus column.
CURLine	curline.0	6
	curline.1	Value of CURLINE setting.
	curline.2	Line number within window of current line.
	curline.3	Contents of the focus line in mixed case.
	curline.4	Status of both the new and change settings for the focus line in the current edit session. "ON" if the line has been altered (new or changed). "OFF" if the focus line has not been altered.
	curline.5	Status of the new and change setting for the focus line in the current edit session. "NEW CHANGED" if the line has been added. "OLD CHANGED" if the line has been changed. "OLD" if the line is not new and unchanged.
	curline.6	Selection level of the focus line.
CURSor	cursor.0	4
	cursor.1	Line number of cursor in CBLe window. 0 if SCALE is ON and cursor is on the scale line.
	cursor.2	Column number of cursor in CBLe window.
	cursor.3	Line number of cursor in file. If the cursor is outside the file window area then a value of -1 is returned.
	cursor.4	Column number of cursor in file. If the cursor is outside the file window area then a value of -1 is returned.

DEFPROFile	`defprofile.0`	1
	`defprofile.1`	Name of default profile macro.
DISPlay	`display.0`	2
	`display.1`	Minimum displayable selection level.
	`display.2`	Maximum displayable selection level.
DSN	`dsn.0`	1
	`dsn.1`	Data Set Name portion of the fileid belonging to the current file.
DSORG	`dsorg.0`	1
	`dsorg.1`	PDS|SEQ|KSDS|ESDS|RRDS|LDS|CMS|LIBR
ENVVars	`envvars.0`	2
	`envvars.1`	ON|OFF
	`envvars.2`	ENVVAR delimiter character.
EOLIn	`eolin.0`	1
	`eolin.1`	CR|LF|NL|CRLF|LFCR|CRNL|X'*string*'
EOLOut	`eolout.0`	1
	`eolout.1`	CR|LF|NL|CRLF|LFCR|CRNL|X'*string*'
FLSCReen	`flscreen.0`	2
	`flscreen.1`	File line number of first text line visible in the window view.
	`flscreen.2`	File line number of last text line visible in the window view.
FMode	`fmode.0`	1
	`fmode.1`	File Mode portion of the fileid belonging to the current file.
FName	`fname.0`	1
	`fname.1`	File Name portion of the fileid belonging to the current file.
FPath	`fpath.0`	1
	`fpath.1`	File Path portion of the fileid belonging to the current file.
FType	`ftype.0`	1
	`ftype.1`	File Type portion of the fileid belonging to the current file.
FIDChanged	`fidchanged.0`	1
	`fidchanged.1`	The current setting of the FIDCHANGED flag, **ON** or **OFF**.
FILEId	`fileid.0`	1
	`fileid.1`	Full fileid of the current file.
HEXSTRING	`hexstring.0`	1
	`hexstring.1`	ON|OFF
HSCROLLCursor	`hscrollcursor.0`	1
	`hscrollcursor.1`	ON|OFF
IMPMACro	`impmacro.0`	1
	`impmacro.1`	ON|OFF
INIFile	`inifile.0`	2
	`inifile.1`	`SYSTEM   system.cbliini.fileid`
	`inifile.2`	`USER     user.cbliini.fileid`
INIVars	`inivar.0` `inivars.0`	Total number of variables set in both the SYSTEM and USER CBLIINI files.
	`inivar.i` `inivars.i`	All specified system and user CBLIINI file variables and their values in alphabetical order. (i=1 to inivar.0) The first token of inivar.i is "SYSTEM" or "USER" indicating the CBLIINI file from which the variable has been extracted. SYSTEM CBLIINI variables are extracted before USER CBLIINI variables. Each CBLIINI variable has 2 levels of naming and is displayed as vlev1.vlev2. • vlev1 identifies the class of the variable as specified in CBLIINI by [nnnnn] or (nnnnn). • vlev2 is the variable name. Both vlev1 and vlev2 are in upper case. The variable name is immediately followed by an "=" (equals) sign and the value in mixed case as supplied in the CBLIINI file. e.g.

```
SYSTEM   CBLVCAT.ESR=222
SYSTEM   EDIT.INSTANCE=Single
USER     EDIT.INITIALSIZE=185,70
```

INSTANCE	instance.0	1
	instance.1	SINGLE|MULTIPLE
INTERFace	interface.0	1
	interface.1	CBLe|ISPF
ISPFMODE	ispfmode.0	1
	ispfmode.1	ON|OFF
KEY	key.0	2
	key.1	Start column of the primary key in the edited KSDS data set.
	key.2	End column of the primary key in the edited KSDS data set.
LASTmsg	lastmsg.0	1
	lastmsg.1	Text of last message displayed (mixed case).
LCOLor	lcolor.0	Number of SET LCOLOR commands in effect for the current file.

	lcolor.i	All SET LCOLOR definitions and their associated parameters in the order in which they were entered (i=1 to lcolor.0).
		
		Each lcolor.i variable contains the following information:
		
		1. The string line-target (as entered on SET LCOLOR.)
		Note that the line-target may contain more than one token. e.g. "word /to/", "/string containing blanks/", etc.
		2. The string line-target (as entered on SET LCOLOR.)
		3. Colour in lower case.
		4. Extended highlighting in lower case.
		5. Name associated with the SET LCOLOR command in upper case. If no name was specified, the target string is used.
		6. "case" in lower case.
		7. Either "respect" or "ignore" in lower case. (The prevailing SET CASE value for string target search when SET LCOLOR was issued.)
		8. "zone" in lower case.
		9. Left zone value. (The prevailing SET ZONE left zone value when SET LCOLOR was issued.
		10. Right zone value. (The prevailing SET ZONE right zone value when SET LCOLOR was issued.
		
		e.g.
		
		`/=AB/ pink default /=AB/ case respect zone 1 1`
		
		`/xYZ/ red revvideo sl1 case ignore zone 8 20`

LCOLour	lcolour.0	Same as LCOLOR.
	lcolour.i	
LENgth	length.0	1
	length.1	Length of focus line.
LIne	line.0	1
	line.1	Line number within current file of focus line.
LINENd	linend.0	2
	linend.1	ON|OFF
	linend.2	Linend character.
LINEFLAG	lineflag.0	3
	lineflag.1	NEW|NONEW
	lineflag.2	CHANGE|NOCHANGE
	lineflag.3	TAG|NOTAG
LISTFILEACTion	listfileaction.0	1
	listfileaction.1	BROWSE|ERASE|NONE
LOADWarning	loadwarning.0	2
	loadwarning.1	ON|OFF
	loadwarning.2	File edit load warning threshold in number of bytes.
LRECL	lrecl.0	2
	lrecl.1	Defined LRECL value of the current file.

	lrecl.2	Length of longest record on load of the current file.
LScreen	lscreen.0	6
	lscreen.1	MDI child window depth (number of rows).
	lscreen.2	MDI child window width (number of columns).
	lscreen.3	MDI child window vertical position within the client area (row number).
	lscreen.4	MDI child window horizontal position within the client area (column number).
	lscreen.5	MDI client area depth (number of rows).
	lscreen.6	MDI client area width (number of columns).
MACROPath	macropath.0	Number of macro path elements set by MACROPATH
	macropath.i	All specified macro path elements in search order. (i=1 to macropath.0).
MBR	mbr.0	1
	mbr.1	PDS Member portion of the fileid belonging to the current file.
MDIList	mdilist.0	1
	mdilist.1	ON|OFF
MSGMode	msgmode.0	1
	msgmode.1	ON|OFF
	msgmode.2	WRAP|NOWRAP
NBWindow	nbwindow.0	3
	nbwindow.1	Number of MDI child window edit views.
	nbwindow.2	Number of edit views of the current file.
	nbwindow.3	Edit view number of the current file.
OPSys	opsys.0	3
	opsys.1	Operating System name.
	opsys.2	Operating System release.
	opsys.3	Reserved (Null string).
PFKey	pfkey.0	24
	pfkey.i	Current PFKey assignments for PFKeys 1-24. (i=1,24).
Point	point.0	0 if focus line is not a named line; otherwise, 1.
	point.1	Line number and names of the focus line, if focus line is named.
Point*	point.0	Number of named lines.
	point.i	Line number and names of the ith named line.
PREfix	prefix.0	3
	prefix.1	ON|OFF|NULLS
	prefix.2	LEFT|RIGHT
	prefix.3	Width of prefix area in number of bytes.
PSCOpe	pscope.0	1
	pscope.1	ALL|DISPLAY
RANGE	range.0	2
	range.1	Line number of first line in range
	range.2	Line number of last line in range. (Equal to one less than RANGE.1 if no lines in current range)
RECFm	recfm.0	1
	recfm.1	FIXED|VARIABLE
RESERved	reserved.0	0 if no reserved lines; otherwise 1.
	reserved.1	List of reserved line numbers, if any.
RING	ring.0	Number of files being edited in the ring.
	ring.i	Information on each file being edited in the ring (i=1 to ring.0).

The tokens returned in ring.i are:

1. Fileid.
2. Line number of current line.
3. Column number of current column.
4. File Size (number of lines.)
5. Alteration count.

e.g.

```
CBL.CMX(NBJ) Line=23 Col=1 Size=429 Alt=0,0
NBJ.TEST Line=0 Col=67 Size=1 Alt=7,7
```

SCOLor

scolor.0 — Number of SET SCOLOR commands in effect for the current file.

scolor.i — All SET SCOLOR definitions and their associated parameters in the order in which they were entered (i=1 to scolor.0).

Each scolor.i variable contains the following information:

1. The string line-target (as entered on SET SCOLOR.)
   Note that the line-target may contain more than one token. e.g. "word /to/", "/string containing blanks/", etc.
2. Colour in lower case.
3. Extended highlighting in lower case.
4. Name associated with the SET SCOLOR command in upper case. If no name was specified, the target string is used.
5. "case" in lower case.
6. Either "respect" or "ignore" in lower case. (The prevailing SET CASE value for string target search when SET SCOLOR was issued.)
7. "zone" in lower case.
8. Left zone value. (The prevailing SET ZONE left zone value when SET SCOLOR was issued.
9. Right zone value. (The prevailing SET ZONE right zone value when SET SCOLOR was issued.

e.g.

```
/</ red default /</ case respect zone 1 *
'IQ00' yellow uscore Q1 case ignore zone 20 40
```

SCOLour

scolour.0 — Same as SCOLOR.

scolour.i

SCOPE

scope.0 — 1

scope.1 — DISPLAY|ALL

SCReen

screen.0 — 2

screen.1 — Depth of 3270 terminal screen.

screen.2 — Width of 3270 terminal screen.

SELect

select.0 — 2

select.1 — Selection level of focus line. (Equivalent to curline.6)

select.2 — Maximum selection level of all lines in current file for the current range.

SHADow

shadow.0 — 1

shadow.1 — ON|OFF

SIze

size.0 — 1

size.1 — Number of lines in the current file.

SIZEWarning

sizewarning.0 — 2

sizewarning.1 — ON|OFF

sizewarning.2 — File edit file size warning threshold in number of bytes.

STAY

stay.0 — 1

stay.1 — ON|OFF

STReam

stream.0 — 1

stream.1 — ON|OFF

SYNonym

synonym.0 — 1

synonym.1 — ON|OFF

SYNonym *

synonym.0 — Number of synonyms defined.

synonym.i — Extracts all defined synonyms in the order in which they were entered. (i=1 to synonym.0).

The format of the tokens returned in synonym.i are:

1. New name.
2. Synonym definition.

THIGHlight

thighlight.0 — 2

thighlight.1 — ON|OFF

thighlight.2 — ALL|FIRST

UNDO	undo.0	3
	undo.1	Number of levels of changes in the current file that may be undone.
	undo.2	Number of levels of changes in the current file that may be redone.
	undo.3	Amount of memory, in kilobytes, containing undo information for current file.
UNDOING	undoing.0	3
	undoing.1	ON|OFF
	undoing.2	The maximum number of undo levels CBLe will attempt to store in memory.
	undoing.3	The maximum amount of memory, in kilobytes, that will be used to store undo information.
USERname	username.0	1
	username.1	The current user's login userid.
VARblank	varblank.0	1
	varblank.1	ON|OFF
VERSION	version.0	4
	version.1	CBLe
	version.2	Version number. (x.xx)
	version.3	Date of build. (yyyy-mm-dd)
	version.4	Time of build. (hh:mm)
WINNAMe	winname.0	4
	winname.1	CBLi window name for the current MDI child edit view.
	winname.2	Title bar contents for the current MDI child edit view.
	winname.3	CBLi window name for the current MDI parent.
	winname.4	Title bar contents for the current MDI parent.
WINPOS	winpos.0	4
	winpos.1	Row number in the client area of the current edit view.
	winpos.2	Column number in the client area of the current edit view.
	winpos.3	Row number in the screen display of the current frame window.
	winpos.4	Column number in the screen display of the current frame window.
WINSIZe	winsize.0	10
	winsize.1	Number of rows in the current edit view.
	winsize.2	Number of columns in the current edit view.
	winsize.3	Number of rows in the current edit view's display area.
	winsize.4	Number of columns in the current edit view's display area.
	winsize.5	Number of rows in the parent window.
	winsize.6	Number of columns in the parent window.
	winsize.7	Number of rows in the parent window's display area.
	winsize.8	Number of columns in the parent window's display area.
	winsize.9	Number of rows in the 3270 terminal.
	winsize.10	Number of columns in the 3270 terminal.
WRap	wrap.0	1
	wrap.1	ON|OFF
Zone	zone.0	2
	zone.1	Left zone column.
	zone.2	Right zone column.

See Also:

QUERY
SET

FILE, FFILE

Syntax:

```
>>-- FILE ---+------------+-------------------------------------------------><
             |            |
             +-- fileid --+
```

```
>>-- FFILE --+-----------+-----------------------------------------------><
             |           |
             +-- fileid --+
```

Description:

Save the current file to disk with an associated fileid and, if successful, exit the edit view and place focus on the previous CBLe edit view in the edit ring.

If a *fileid* is specified, the data is saved under the new fileid. File data that exists on disk under the original fileid is unchanged.

If *fileid* is not specified, FILE attempts to write the file to disk using the currently assigned fileid. By default, this fileid is that which was used to initially edit the file, unless subsequently updated by a SET FILEID, FNAME, FTYPE, FMODE, FPATH command.

If the fileid to be used by FILE does not already exist, a new file will be created. For MVS Sequential, PDS(E) and VSAm data sets, this will open the Allocate NonVSAM or Define VSAM dialog window prompting the user to provide the required new data set characteristics.

The file save will fail and return an error if either of the following conditions are true:

- The fileid used to save the data differs from that used on the initial edit of the file and this fileid already exists for a file on disk.
- For HFS files, the current "Modified" timestamp of the file is later than the time at which the file was last saved, or else first edited, in the current CBLe session. i.e. the file's data has been changed by some other process or user edit.

FFILwill save the file regardless of the above error conditions.

Where *fileid* is identified as being an HFS path (i.e. begins with "." or contains "/"), DSORG is set to be HFS and the file is saved with the permission mode 740 and the following record delimitation:

- If the current RECFM is F, the HFS file will contain no EOL delimiters and each record will be padded to the current LRECL value.
- For RECFM V or U, HFS file records will be delimited with EOL (end-of-line) characters as defined by the current setting of EOLOUT.

Parameters:

fileid

The fileid to be assigned to the file on writing it to disk.

For MVS data sets, *fileid* may be the DSN of a Sequential or VSAM data set, the DSN and member name of a PDS(E) library member or an HFS absolute or relative path name.

For VSE files, *fileid* may be the full LIBR member id, lib.sublib.mname.mtype.

For CMS files, *fileid* may be the full CMS file id, FNAME FTYPE FMODE (or FNAME.FTYPE.FMODE) If only two qualifiers are specified, FMODE defaults to the current FMODE, and if only one qualifier is specified, then FTYPE and FMODE default to the current FTYPE and FMODE.

See Also:

SET FILEID
SAVE
QUIT

FILLBOX

Syntax:

```
>>-- FILLbox --+------------+--><
               |            |
               +---- x -----+
               |            |
               +-- string --+
```

Description:

Fill a marked block with the specified single character or insert the specified text string in every line of the marked block beginning at the left column of the block.

The original contents of the block are overwritten.

Where the block is a line block, only positions between the left zone column and the right zone column are filled.

Parameters:

`x`	A single fill character for the block. All positions in the block will contain this character. If no character is specified, the block is filled with blanks.
`string`	Text string to be placed in the left column of every line in the block. Where more than 1 blank separates the text string from the command verb, the additional separating blanks are treated as part of the text string. The specified string is truncated or padded with blanks accordingly, in order to fit the marked block.

Example:

`fill`	Fill a marked block with blanks.
`fill A`	Fill a marked block with character "A".
`fill ABC`	Fill each line of a block with "ABC" in the first marked column and pad with blanks.
`fill  XY Z`	Fill each line of a block with " XY Z" in the first marked column and pad with blanks.

FILLDIALOG

Syntax:

```
>>-- FILLDialog -----------><
```

Description:

Open the CBLe Fill Dialog Window to perform a FILLBOX command.

This dialog window may also be opened by selecting **Fill** from the **Actions** menu item in the CBLe Main Menu Bar.

Parameters:

None.

See Also:

CBLe Fill Dialog Window

FIND, FINDUP

Syntax:

```
>>----- Find --------+----------+------><
                     |          |
                     +- string -+

>>--+-- FINDUP --+---+----------+------><
    |            |   |          |
    +-- FUP -----+   +- string -+
```

Description:

FIND is also an ISPF primary command.
If INTERFACE=ISPF is in effect (default for CBLi in an MVS environment), then the ISPF version of the command is executed instead. See section ISPF/CBLe CLI Command Precedence and IBM publication "ISPF: Edit and Edit Macros".
Use the CBLe CLI command ECOMMAND to override.

Find the first line following (FIND) or previous to (FINDUP) the focus line that contains the specified string in column 1 and make this line the new focus line.

Where more than 1 blank separates the text string from the command verb, the additional separating blanks are treated as being part of the text string.

If WRAP is set ON and string is not found before bottom-of-file or top-of-file is encountered, then the scan continues from the opposite extreme of the file up to and including the focus line.

Parameters:

`string` Search text string.
No delimitting characters should be specified.

Blanks in the search text are each treated as a wildcard representing a single character in the file data. In order to find a blank space, the "_" (underscore) character should be specified in the corresponding position of the find string.

If string is not specified, then the string argument specified on the last FIND/FINDUP command executed, is used. If no previous FIND/FINDUP command has been executed in this instance of CBLe, then the following message is displayed:

```
EDT060E No remembered operand for FIND command.
```

Example:

`find Hello` Focus line is set at the first line below the focus line to contain the string "Hello" starting in column 1.

`f ABC DEF` Focus line is set at the first line below the focus line to contain any character in column 1, the string "ABC" starting in column 2, any two characters in columns 5 and 6, and the string "DEF" starting in column 7.

`fup _XYZ` Focus line is set at the first line above the focus line to contain a blank in column 1 followed by the string "XYZ" in column 2.

See Also:

NFIND, NFINDUP
SET WRAP
SET CASE

FORWARD

Syntax:

```
>>-- FOrward ----------------------><
```

Description:

The FORWARD command scrolls the focus window forwards 1 page towards the bottom of the file.

Parameters:

None.

See Also:

BACKWARD

FREE

Syntax:

```
>>-- FREE -- freeparms ----><
```

Description:

The FREE command may be used to:

1. Dynamically unallocate a single data set.

2. Override the disposition of allocated data sets.
3. Override the output class.

FREE allows users to unallocate files whether or not a TSO environment is available. The syntax of the command closely matches that of the TSO FREE command and so most FREE commands, executed without TSO as a prefix, will give the same results.

FREE is identical to the ALLOCATE -FREE command and is supported for MVS only.

Parameters:

`freeparms` Parameters supported by the TSO FREE command as follow:

`DDname(ddname) or File(ddname), DSName('dsn') or DAtaset('dsn'), KEEP, DELETE, CATALOG, UNCATALOG, SYSOUT(class) and SPIN(UNALLOC)`

Examples:

`free f(indd)`
Unallocate an existing data set.

`free f(ixnbj) keep`
Unallocate a ddname overriding the disposition with KEEP.

See Also:

ALLOCATE

GET

Syntax:

```
                          +-- 1 -------------- * -----------+
                          |                                 |
>>-- GET ---- fileid ----+---------------------------------+----------------->< 
                          |                                 |
                          |                +-- * --------+  |
                          |                |             |  |
                          +-- start_line --+-------------+--+
                                           |             |
                                           +-- n_lines --+
```

Description:

The GET command reads lines from a specified file and inserts them into the current file after the focus line. Following execution of GET, the last of the inserted lines becomes the focus line.

GET is similar to the INTERFACE=ISPF command COPY which is in effect by default for when CBLi executes in an MVS environment. See IBM publication "ISPF: Edit and Edit Macros" for use of the ISPF COPY primary command.

Where *fileid* is identified as being an HFS path (i.e. begins with "." or contains "/"), records, and so record line numbers, are determined based on the EOL format in the HFS file's directory entry. If this EOL format is is not defined, records are determined by the prevailing setting of EOLIN for the current file.

Parameters:

`fileid`
: The fileid of the file containing the lines of data to be inserted.
The contents of this file are unaffected by the GET command.

`start_line`
: The line number of the first line to be retrieved from the file *fileid* for insertion into the current file. Subsequent lines will be retrieved following this line number.
Default is line number 1.

`n_lines`
: The number of lines in total to be retrieved from the file *fileid* for insertion into the current file. "*" (asterisk) indicates that all lines, starting at line number specified by *start_line* up to and including the last line of the file, will be retrieved and inserted in to the current file.
Default is "*" (asterisk).

Example:

`GET CBL.CBLI.INI`
Get all lines of the MVS sequential data set and insert them into the current file after the focus line.

`GET CBL.SSC.CTL(DIRD01)  8 10`
Get lines 8 to 17 (inclusive) of the MVS PDS member and insert them into the current file after the focus line.

`GET CBLNAME ASSEMBLE A   16 *`
Get all lines, starting at line number 16 of the CMS file, and insert them into the current file after the focus line.

`GET PRD2.CBL210.CBLIINST.HTML 22`
Get all lines, starting at line number 22 of the VSE LIBR member, and insert them into the current file after the focus line.

`GET ../tmp/u2009_06_01.SHCMDDES.ls.txt`
Get all lines from the specified HFS file and insert them into the current file after the focus line.

HELP

Syntax:

```
>>-- Help --+-----------+--><
            |           |
            +-- topic --+
```

Description:

Use the HELP command to open the Help Window and optionally link directly to help on a specific CLI command.

Where topic is not specified or not found, the relevant table of contents is displayed.

The Help window may also be opened via the Help item of the window's menu bar.

Parameters:

Parameters:

`topic` Display help on a specific topic.
If topic is enclosed in single or double quotes, the string is treated as the fileid of an HTML data set to be browsed. This may be the fully qualified fileid of an HTML document or the name of a PDS member that exists in the defualt HELP library.

Example:

`help fill`	Display the help page for the FILL command.
`H "OEM.CBL.HTML(TEST)`	Open a specific HTML document library member.
`H "WINSIZEW"`	Open the CBLi Help member name WINSIZEW.

ICOMMAND

Syntax:

```
>>-- ICommand -- command -------><
```

Description:

Where a command exists in both the CBLE and ISPF edit command sets and INTERFACE=CBLE is active, ICOMMAND may be used as a prefix in order to force the use of the ISPF version of the command.

If INTERFACE=ISPF is active, the ICOMMAND prefix has no effect.

Parameters:

`command`	Any CBLe command followed by its parameters.

Example:

```
ICommand Change abc 'DEF' all
```
The ISPF version of the CHANGE command is used regardless of INTERFACE=CBLE.

See Also:

ECOMMAND

IMMEDIATE

Syntax:

```
>>-- IMMediate -- macrodef ----------><
```

Description:

IMMEDIATE allows the user to write and execute a short CBLe REXX macro from the command line.

IMMEDIATE is especially useful for executing one-off functions and for testing excerpts of logic when writing CBLe macros.

Parameters:

`macrodef`	A short CBLe REXX macro that may be defined on a single line.

Example:

```
immediate  'msg' x2d(1AE)
```

Convert x'1AE' to decimal and display the result on the message line.

```
imm 'wrap off'; ':1'; do forever; 'nomsg /=START=/';if rc 0 then leave; 'cl:12';'cov **';'-1';'a5';'5';end;':1'
```

Locate all lines containing the string "=START=", overlay column 12 with '**' and insert 5 blank lines prior.

See Also:

DEFINE
PURGE

INPUT

Syntax:

```
>>--+-- Input ---+--+------------+--><
    |            |  |            |
    +-- Insert --+  +-- string --+
```

Description:

Insert a new line following the focus line containing the specified text string. The new line becomes the focus line and the cursor placed in column 1 of the new line.

Where no string is specified, the inserted line is blank. This is the same as ADD.

Parameters:

`string`	Text string to be inserted in column 1 of the new line. block. Where more than 1 blank separates the text string from the command verb, the additional separating blanks are treated as part of the text string.

Example:

`input/Hello/`	Insert a line containing the text string "/Hello/" following the focus line.
`ins  ABC`	Insert a line containing the text string " ABC" following the focus line.

JOIN

Syntax:

```
>>-- Join --+-------------+--><
            |             |
            +-- ALigned --+
```

Description:

Join text from the line below the focus line to the focus line itself starting at the focus column. Text at, and to the right of, the focus column is overlayed.

The LRECL setting defines the last column of the text to be joined.

Where ALIGNED is not specified, text starting in column 1 of the line below the focus line is joined to the focus line.

Parameters:

`ALIGNED`	A number of leading blanks, not exceeding the number of leading blanks in the focus line, are stripped from the start of the line below the focus line before the join is actioned.

Example:

In the following, focus column is column 13.

```
Command>

<...+....1..|.+....2....+....3....+....4....+
      Hello Dave
   World
```

```
Command> join

<...+....1..|.+....2....+....3....+....4....+
      Hello    World
```

```
Command> join al

<...+....1..|.+....2....+....3....+....4....+
      Hello World
```

See Also:

SPLIT
SPLTJOIN

LEFT

Syntax:

```
           +- 1 ----+
           |        |
>>-- LEft -+--------+--><
           |        |
           +- n ----+
           |        |
           +- HALF -+
```

Description:

LEFT is also an ISPF primary command.
If INTERFACE=ISPF is in effect (default for CBLi in an MVS environment), then the ISPF version of the command is executed instead. See section ISPF/CBLe CLI Command Precedence and IBM publication "ISPF: Edit and Edit Macros".
Use the CBLe CLI command ECOMMAND to override.

LEFT positions the focus column one or more columns to the left of the current focus column. Alternatively, position the focus column to the left of the current focus column, for a number of columns equal to one half the width of the edit view.

Where the specified number is greater than the number of columns to the left of the focus column, then column 1 becomes the focus column.

Parameters:

`n` The number of columns to the left of the current focus column at which the new focus column is to be set. If omitted this parameter defaults to 1.

`HALF` Position the focus column a number of columns equal to half the width of the edit view, from the current focus column position.

Example:

`left`	Set the focus column to be 1 column to the left of the current focus column.
`le10`	Set the focus column to be 10 columns to the left of the current focus column.

See Also:

RIGHT

LESS

Syntax:

```
>>-- LESS --+---------+-- line-target --><
            |         |
            +-- TAG --+
```

Description:

Remove from the window view or un-tag all lines that match line-target.

If the current DISPLAY setting is DISPLAY 0 1 or DISPLAY 1 1, CBLe assumes that the ALL command has previously been issued, thus allocating a subset of lines within the file with a selection level of 1. In this case the selection level of all lines matching the line-target are set back to 0.

If any other DISPLAY setting is in effect, CBLe sets DISPLAY 1 1, sets the selection level of all lines that match the line-target to 0 and sets the selection level of all other lines to 1.

If the TAG or MORE TAG command has been issued, the tag bit may be set on for specific lines in the display. Lines with the tag bit set on are automatically highlighted.

The TAG parameter may be inserted before the line-target of a LESS command to remove the tag bit from lines in the display that satisfy the specified line-target.

Parameters:

`TAG`	Indicates un-tagging required as opposed to excluding lines from the display.
`line-target`	line-target condition.

Examples:

`less /XYZ/`
All lines containing string "XYZ" are excluded from the window view.

`less tag /###/`
Remove the tag bit and highlight from lines containing the string "###".

See Also:

ALL
TAG
MORE

LIST

Syntax:

```
>>-- LIst -- listtype -- /listparms/ --+---------------------+--+---------+-->
                                       |                     |  |         |
                                       +- STEM rexx_stemvar -+  +- STRIP -+
                                       |                     |
                                       +- FILE filename -----+


    +- Lines ---+
    |           |
 >--+-----------+--+-------------------------+--+-----------+-+-----------+-><
    |           |  |                         |  |           | |           |
    +- Columns -+  +- SUBset /select_clause/ -+  +- CASEIgn -+ +- RECURSE -+
```

Description:

Extract rows of data returned from various CBLi LIST type commands and either place the output in a temporary edit view or assign the fields to REXX stem variables for use in a CBLe REXX macro.

Parameters:

listtype
The CBLi list type function to extract. This must be one of the following:

AMS	List IDCAMS command output.
APE	List Assembler Program Environment. This is the list of modules in the current version of CBLi.
FS FSEARCH	List records that match a search string.
HFS PATH	List HFS files.
LA ALLOCated	List allocated datasets.
LC CATalog	List catalog entries.
LD DATasets DSNs FILes	List dataset attributes.
LJQ JOBENQueues	List MVS enqueues for a given job.
LL LIBrary	List library members.
LQ ENQueues	List MVS enqueues.
LV VTOC	List VTOC entries.

`LVOL` `VOLumes`	List DASD volumes.
`LVR`	List CBLVCAT Raw output.
`LX` `EXTents`	List VTOC extents.
`POWER`	VSE only: POWER Queue list output.
`RETRIEVE`	The list of commands available via the RETRIEVE command, which, by default, is assigned to PF12.
`STRUCTure`	MVS only: Structured Data Environment DISPLAY STRUCTURE list output.
`SWA`	List Attributes of all open windows.

See the relevant List type commands for details of the column names and data returned for each.

`listparm`

A parameter string passed to the list function. This string must always be present. If it contains no blanks or special characters it can be blank-delimited, otherwise it must be delimited by a pair of special characters. e.g. If a null string is required it can be provided as "//".

`FILE filename`

A keyword parameter which requests that the LIst command places the listed data in a file for the user to edit. *filename* is the name used for the edit view but the data is not written to disk. The user may file the data if required.

This option is the default if the LIst command is issued from the command line or via a function key in which case the *filename* will be "%user%.*listtype*.LIST".

`STEM rexx_stemvar`

A keyword parameter which requests that the LIst command places the listed data in an array of REXX variables with this stem name.

This option is the default if the LIst command is issued from a macro.

If the stem name is not given then the list type name is used.

If the stem name does not end with a period then one is added.

`STRIP`

A keyword parameter which requests that the data is stripped of leading and trailing blanks before being placed in REXX variables.

`Lines`

A keyword parameter which requests that the LIst command returns complete list lines.

If the FILE option is used then the list column header and list lines are place in an edit window in the current ring with the file name specified (but not actually written to disk).

If the STEM option is used then the following variables are set:

`stem.0`	The number of list lines.
`stem.i`	The ith list line.
`stem.columns`	The list column header line.

`Columns`

A keyword parameter which requests that the LIst command returns the list data in column variables.

If the FILE option is used, or the LIst command is not issued from a macro, this option is invalid. An error message is issued and no data is returned.

If the STEM option is used then the following variables are set:

`stem.0`	The number of list columns.
`stem.i.colname`	The value of column colname for the ith list line.
`stem.columns.0`	The number of columns.
`stem.columns.j`	The name of the jth column.

`SUBset /clause/`

A keyword parameter which must be followed by a list subset command delimited by a special character.

The subset *clause* can contain one or more of the following (in any order):

`select clause`	Use the select clause to define the list of column names to return. The default is "*" which means all columns.
`where clause`	Use the where clause to apply a filter which restricts the lines returned based on a condition defined in terms of the list's column names and their values.

sort (order by) clause	values in the columns of the list. Use the sort or order by clause to determine the order in which the list lines are returned based on column values.

See Selecting, Sorting and Filtering for details of the syntax of these clauses.

CASEIgn

For **HFS lists** only, bypass case sensitivity for the **name** portion of the HFS path specified in *listparms*. The name portion of the HFS path is the character string at the end of the path that follows the last "/" (slash) of the fileid, or is the entire path name if "/" is not specified.

RECURSE

For **HFS lists** only, recursively list the contents of all sub-directories found within the *listparms* HFS path specification.

Examples:

```
list volumes/CBL*/ file lac.vollist subset/select unit,vol,freecyl/
li alloc /SYSEXEC/
li alloc SYSPROC
li library/cbl.cbli120.asm(edt*)/ stem ll.
li library/cbl.cbli120.asm(edt*)/ file lac.liblist
li catalog/cbl.cbli*.**/ file lac.catlist
li dataset/cbl.cbli*.**/ file lac.dsnlist
li vtoc /cblmct/ file lac.vtoclist subset / select vol,dsn,org /
li extent /cblmct/ file lac.extlist
li enqueue/sysdsn LAC./ file lac.enqlist
li enqueue/spfedit LAC./ file lac.enqlist
```

LOCATE

Syntax:

- **Interface CBLe**

```
>>-- Locate --+--------------------------+---------------------------------><
              |                          |
              +- line-target -+-----------+
                              |          |
                              +- command -+
```

- **Interface ISPF**

```
>>-- Locate --+-- line_num -------------------------------------------------+-><
              |                                                      |
              +-- .label -----------------------------------------------+
              |                                                      |
              | +-- NEXT --+                  +-- .ZFIRST -- .ZLAST ---+ |
              | |          |                  |                      | |
              +-+----------+--+-- Change ----+--+----------------------+-+
                |          |  |            |  |                      |
                +-- FIRST -+  +-- COmmand ---+  +-- .labela -- .labelb --+
                |          |  |            |
                +-- LAST --+  +-- ERror -----+
                |          |  |            |
                +-- PREV --+  +-- EXcluded --+
                              |            |
                              +-- Label -----+
```

Description:

Locate the line within the displayed data that matches the locate search criteria.

If INTERFACE=ISPF is in effect (default for CBLi in an MVS environment), then the ISPF interface version of **LOCATE** is executed. CBLe CLI command ECOMMAND may be used to force use if the CBLe interface version of LOCATE and command ICOMMAND may be used to force use if the ISPF interface version.

See section ISPF/CBLe CLI Command Precedence for details of CLI command format based on the CBLe text edit environment, and also the IBM publication "ISPF Edit and Edit Macros" for details on use of the ISPF Edit primary command LOCATE. The remainder of this section deals with the CBLe interface version only.

Locate the first line from the focus line to contain the specified *line-target* and make this line the new focus line. If the cursor is outside the display area (e.g. on the command line) or the first line matching the line target falls outside the current displayed page of data, the display is scrolled so that the new focus line also becomes the current line of the display. Otherwise, the cursor is simply relocated to the same column within the new focus line.

The command verb LOCATE is optional where line-target begins with a non-alpha character.

If WRAP is set ON and the line-target is not found before bottom-of-file or top-of-file is encountered, then the scan continues from the opposite extreme of the file up to and including the focus line.

Another CBLe command may immediately follow the line-target specification on LOCATE command. The command is executed after the target line has been made the focus line.

Where line-target is not specified, CBLe repeats the last LOCATE command issued.

Parameters:

`line-target`
: Any line-target condition.

`command`
: Any CBLe command.

Example:

`l3`
: Focus line is set 3 lines down from the current focus line.

`:50`
: Focus line is set at line 50.

`/Hello/`
: Focus line is set at the first line below the focus line to contain the string "Hello".

`-*`
: Focus line is set to the top-of-file line.

`:10 ADD 3`
: Focus line is set at line 10 then 3 lines are added.

`:6 /CBL/ 1 DEL`
: Focus line is set at line 6, then to the first line below line 6 to contain the text "CBL", then to 1 line below the line containing "CBL" before eventually deleting the focus line.

LOWERCASE

Syntax:

```
>>-- LOWercase -----+----------------+--><
                    |                |
                    +- group-target -+
```

Description:

Lower case all alpha characters in the target area.

Only alpha characters that are within the current ZONE columns will be lower cased.

Where BLOCK is specified as the group-target and the target area is a box block, then the ZONE setting is ignored and all alpha characters within the block are lower cased.

The focus line is not changed by a LOWERCASE command.

Parameters:

`group-target`
: Group-target condition defining the end of the source target area. If the group-target is not satisfied then the LOWERCASE command will fail.

Example:

`lowercase 6`	Alpha characters in the focus line and the 5 lines following are lower cased.
`low -8`	Alpha characters in the focus line and the 7 lines preceding are lower cased.
`lower /SELC/`	Alpha characters in the focus line and all lines up to, but not including, the first line following the focus line to contain the string "SELC", are lower cased.

See Also:

UPPERCASE

MACRO

Syntax:

```
>>-- MACRO -- macroname --+--------+--><
                          |        |
                          +- text -+
```

Description:

Execute the REXX language macro specified by macroname. Any text specified following the macroname is passed to the macro as an argument. The specified macro is read into memory from disk, executed and removed from memory on completion.

Where IMPMACRO is set ON (default), CBLe first checks a user supplied token for a recognised CBLe command. If it is not recognised as a command, CBLe attempts to find a macro with the same name. Therefore, when invoking a macro, the command verb MACRO may not be necessary.

If the macro cannot be located, an error message is issued and the MACRO command fails.

Parameters:

`macroname` Name of the macro to be executed.

If macroname is a full fileid containing file name, path, etc., then CBLe reads the macro from the specified location. If only a file name is specified, CBLe searches each directory in the macro path for a matching macroname.

`text` Text to be passed to the macro as arguments.

Example:

`macro open`	Execute the user macro OPEN, not the CBLe command OPEN.
`ldiff`	Execute the user macro LDIFF. (CBLe command LDIFF does not exist.)

MARK

Syntax:

```
>>-- MARK --+-- Line --+--><
            |          |
            +-- Box ---+
```

Description:

Mark the boundaries of a line or box. MARK LINE marks the focus line as one edge of a line block and MARK BOX marks the focus column in the focus line as one corner of a box block. MARK is most often used in CBLe macros prior to a FILL, DUPLICATE, DELETE, etc. command.

A marked block in a file other than the current file, will be reset when the MARK command is issued.

The marked blocks remain marked until explicitly unmarked. When defining a new boundary for an existing block, the block will resize from the focus line or column to the most extreme boundary of the current block.

Parameters:

`LINE`	Mark the focus line as a boundary for a line block.
`BOX`	Mark the focus column in the focus line as a boundary for a box block.

See Also:

RESET

MORE

Syntax:

```
>>-- MORE --+---------+-- line-target --><
            |         |
            +-- TAG --+
```

Description:

Add to the current window view or tag all lines that match line-target. The MORE command is usually used following an ALL or LESS command.

MORE sets the selection level of all lines matching line-target to selection level 1.

Where the ALL command has not previously been issued, The MORE command also sets DISPLAY 1 1. This is the same as executing the ALL line-target command.

The TAG parameter may be inserted before the line-target of a MORE command to set the tag bit on lines in the display that satisfy the specified line-target. Unlike the TAG command, MORE TAG does not remove the tag bit for lines that do not satisfy the line target.

Parameters:

`line-target`	line-target condition.
`TAG`	Indicates tagging required as opposed to adding extra lines to the display.

Examples:

`more tag /##/`
Tag (highlight) lines that contain the string "##". (Lines that are already have the tag bit on are unaffected.)

`more /XYZ/`
All lines containing string "XYZ", which have been excluded from the window view by a previous ALL or LESS command, are included in the current window view.

See Also:

ALL
TAG
LESS

MOVE

Syntax:

```
>>-- Move -- group-target --+---------------+--><
                            |               |
                            +- line-target -+
```

Description:

MOVE is also an ISPF primary command.
If INTERFACE=ISPF is in effect (default for CBLi in an MVS environment), then the ISPF version of the command is executed instead. See section ISPF/CBLe CLI Command Precedence and IBM publication "ISPF: Edit and Edit Macros".
Use the CBLe CLI command ECOMMAND to override.

Move text from a target area to a target line.

Where BLOCK is specified, MOVE supports moving text between files. Otherwise MOVE can only operate on lines in the same edited file. The block remains marked in its new position and the first line of the moved block becomes the focus line.

Where BLOCK is not specified as the group-target, the last line of the moved target group of lines becomes the focus line.

Where line-target is omitted, the current focus line is used.

Parameters:

`group-target`	Group-target condition defining the end of the source target area. If the group-target is not satisfied then the MOVE command will fail. If BLOCK is specified, then the marked block will be moved as follows: Line Block Marked line(s) are moved to the line immediately following the line-target. Box Block Marked box is moved to the focus column of the target line. **Note:** group-target key word ALL is not supported.
`line-target`	Line-target condition defining a destination target line. If the line-target is not satisfied then the MOVE command will fail. Where the source target area is not a box block, lines are moved to the line immediately following the target line.

Example:

`move 6 :16`	The focus line and the 5 lines following are moved to line below line 16.
`m -8 22`	The focus line and the 7 lines preceding are moved after 22nd line following the focus line.
`mo 3 -/SELC/`	The focus line and the 2 lines following are moved to the line below the first line containing the string "SELC", scanning backwards from the focus line.
`mov block`	The marked box block is moved to the focus column of the focus line and lines that follow up to the depth of the marked block.

See Also:

COPY

MSG

Syntax:

```
>>-- MSG --+------------+---->< 
           |            |
           +-- string --+
```

Description:

Display a message string on the message line. MSG is most often used in CBLe macros.

If no text string is specified, then the message line is cleared.

Parameters:

`string`	Text string to be displayed on the message line.

Example:

`msg 'Hello World'`	Output "hello World" to the message line.

See Also:

EMSG
NOMSG

NFIND, NFINDUP

Syntax:

```
>>----- NFind --------+----------+------><
                      |          |
                      +- string -+

>>--+-- NFINDUP --+---+----------+------><
    |             |   |          |
    +-- NFUP -----+   +- string -+
```

Description:

Find the first line following (NFIND) or previous to (NFINDUP) the focus line that does **not** contain the specified string in column 1 and make this line the new focus line.

Where more than 1 blank separates the text string from the command verb, the additional separating blanks are treated as being part of the text string.

If WRAP is set ON and string is found in column 1 of every line before bottom-of-file or top-of-file is encountered, then the scan continues from the opposite extreme of the file up to and including the focus line.

Parameters:

`string`

Search text string.
No delimitting characters should be specified.

Blanks in the search text are each treated as a wildcard representing a single character in the file data. In order to find a blank space, the "_" (underscore) character should be specified in the corresponding position of the find string.

If string is not specified, then the string argument specified on the last NFIND/NFINDUP command executed, is used. If no previous NFIND/NFINDUP command has been executed in this instance of CBLe, then the following message is displayed:

```
EDT060E No remembered operand for NFIND command.
```

Example:

`nfind Hello` Focus line is set at the first line below the focus line that does **not** contain the string "Hello" in column 1.

`nf  ABC  DEF` Focus line is set at the first line below the focus line **not** to contain the string "ABC" in column 2 and the string "DEF" in column 7.

`nfup _XYZ` Focus line is set at the first line above the focus line **not** to contain a blank in column 1 followed by the string "XYZ" in column 2.

See Also:

FIND, FINDUP
SET WRAP
SET CASE

NOMSG

Syntax:

```
>>-- NOMsg --- command ----------------------------------------------------><
```

Description:

Execute the specified CBLe or SDE command, suppressing the display of any messages that may be returned. NOMSG is most often used in CBLe and SDE edit macros.

Execution of NOMSG stores the current MSGMODE setting, sets MSGMODE OFF, executes the specified command and finally restores the MSGMODE setting.

Note: QUERY and EXTRACT LASTMSG recalls the last message that would have been displayed if the command had been executed without NOMSG.

Parameters:

command
CBLe or SDE command.

Example:

`nomsg  ecommand  change /ABC/DEF/* *`
Any messages generated by the CHANGE command are suppressed.

See Also:

MSG
EMSG

NORECALL

Syntax:

```
>>-- NORecall --- command -->< 
```

Description:

Execute the specified CBLe command but exclude it from CBLe's stack of commands entered. NORecall is most often used in macros.

The specified command will not be recalled when the CBLi function "RETRIEVE -" is executed (Default action for PF10).

Parameters:

command	CBLe command.

Example:

nor change /ABC/DEF/* *	Exclude this execution of CHANGE from the recallable command stack.

OPEN

Syntax:

```
>>--- OPEN -----+------------+---------------------------------------------><
                |            |
                +-- filter --+
```

Description:

Open the OPEN dialog window. This command is equivalent to selecting "open" from the "File" menu on the CBLe window menu bar.

If *filter* is not specified, the Open window displays the list of fileids for the filter mask last specified in an Open window.

Parameters:

filter
Display only files that match the specified file mask filter. The filter supports the following wild cards:

(**Note:** for CMS, valid qualifiers may be considered to be File Name, Type or Mode.)

`*` A single asterisk indicates that either a qualifier or one or more characters within a qualifier can occupy that position. An asterisk can precede or follow a set of characters.

`**` A double asterisk indicates that zero or more qualifiers can occupy that position. A double asterisk cannot precede or follow any characters; it must be preceded or followed by either a dot or a blank.

`%` A single percent sign indicates that exactly one character can occupy that position. (Up to 8 percent signs can be specified in each qualifier.)

On MVS or VSE, an entry field that does not contain an * (asterisk) wild card will be appended with *.** to list those cataloged data sets whose names begin with the entry string.

This parameter is placed in the Filter field of the Open window. If filter is not specified, then the filter mask used is that specified in the last Open window opened. If an Open window has not already been opened for the current instance of CBLe, then the fileid of the current display window is used.

Example:

`open CBL.S`
Open the OPEN window, listing all files whose fileids match the filter mask "CBL.S*.**".

`open SS* CTL *`
Open the OPEN window, listing all CMS files whose fileids match the FN FT FM mask "SS* CTL *".

OPTIONS

Syntax:

```
                +-- ALL -----+
                |            |
>>-- OPTions --+------------+--><
                |            |
                +-- Edit ----+
```

Description:

Excute the OPTIONS command to perform one of the following:

- Execute the QUERY command for all possible query parameters with the exception of QUERY MACRO, to display all current settings.
- Generate a CMX file containing SET commands for all current SET option values. The CMX command file also contains references to HELP pages for each SET option.

The OPTIONS command is also executed via the Options CBLe Menu bar item.

Parameters:

`ALL` Display current query option values.

`EDIT` Generate temporary OPTIONS CMX file containing all available SET commands with prevailing option settings.

OVERLAY

Syntax:

```
>>-- Overlay --- string ---------------------><
```

Description:

Overlay text in the focus line with the specified text string starting at column 1. The text string begins immediately after the single separating blank that follows the OVERLAY command verb.

Characters in the focus line that correspond to blanks in the supplied overlay string are unchanged. In order to overlay a character with a blank, the "_" (underscore) character should be specified in the corresponding position of the overlay string.

All other characters supplied in the overlay string replace characters in corresponding positions in the focus line.

Parameters:

`string` Overlay text string.
Where more than 1 blank separates the text string from the command verb, the additional separating blanks are treated as part of the text string.

Examples:

`overlay abc`
Overlay text at column 1 of the focus line with "abc".

`O ABC_X YZ`
Where original text at column 1 of the focus line is: 0123456789
After the OVERLAY command text is: 0ABC X6YZ9

OVERLAYBOX

Syntax:

```
>>-- OVERLAYBox ----------><
```

Description:

Overlay text on and, if necessary, below the focus line with text from a marked line block or box block.

Where a box block has been marked, text at, and to the right of the focus column is overlayed.

Parameters:

None.

POPUP

Syntax:

```
                         +-----------------+
                         |                 |
                         v                 |
>>-- POPUP ------ / --- string -- / --+------><
```

Description:

For use in CBLe macros only, opens a pop-up menu containing any number of items specified by separate string arguments.

The "/" (slash) character is normally used as the delimiter encompassing and separating each item string in the pop-up menu. However, any non-alphanumeric character that does not have a special meaning to CBLe may be used as a string delimiter (e.g. "%", "#", "."). The delimiter character used must not appear in any of the string arguments.

The following REXX stem variables, which do not require an EXTRACT command, are set on completion of the POPUP command:

`POPUP.0` 1

`POPUP.1` The contents of the menu item selected by the user. This value is returned exactly as specified on the POPUP command, and includes any blanks present in the original string. If no item was selected (because a 3270 AID other than <Enter> was received or because no valid entry was selected), this variable is set to the null string.

Parameters:

`string` Character string of maimum length 64 and enclosed by delimiters to be displayed as a menu item in the pop-up window.
Any number of menu item character strings may be specified, each having a maximum length of 64 characters.

string may be prefixed by "~" (tilde) to represent an item that may not be selected by the user. This item will be

displayed in a different colour (default is blue) to the live menu items (default is white).

If **string** consists only of "**-**" (minus), the menu item is displayed as a separator line within the menu.

Example:

```
'popup /John Smith/Robert Jones/-/Susan Fisher/~Jane Dough/Rachel Meredith/-/New contact/'
if popup.1 = ''  then 'msg No contact name selected.'
else if popup.1 = 'New contact'  then call GetNewName
else 'msg 'popup.1 'selected...'
```

See Also:

DIALOG

PRESERVE

Syntax:

```
>>-- PREServe ------------><
```

Description:

Save current values of SET options so that they may be temporarily changed and ultimately restored using the RESTORE command. PRESERVE is most often used in macros.

SET options are saved for the following:

ARBCHAR, CASE, COLOUR, DISPLAY, IMPMACRO, LINEND, LRECL, MSGLINE, MSGMODE, PREFIX, RECFM, SCALE, SCOPE, SHADOW, STAY, STREAM, UNDOING, VARBLANK, WRAP, ZONE.

Parameters:

None.

See Also:

RESTORE

PURGE

Syntax:

```
                  +----------------+
                  |                |
                  v                |
>>-- PURge ---+-- macroname --+--><
```

Description:

PURGE is used to unload macro definitions that have been loaded into storage via the DEFINE command.

Multiple macros may be unload in a single PURGE execution by specifying a blank separated list of macro names.

Parameters:

`macroname`	The name of the macro in storage.

Example:

`purge puttime delsame addnext`	Unload macros "puttime", "delsame" and "addnext" from storage.

See Also:

DEFINE
IMMEDIATE

QUERY

Syntax:

```
>>-- Query -- option -------------------------------------------------------><
```

Description:

Query information about the current edit environment for the specified option. The information is displayed on the message line.

Parameters:

option
The option for which information is returned.

Currently supported QUERY options are as follow:

Alt	Displays the number of alterations made to the current file since the last AUTOSAVE or SAVE and the number of alterations made to the current file since the last SAVE.
ARBchar	Displays the current setting of ARBCHAR (ON or OFF) including the first and second ARBCHAR characters.
BEEP	Displays the current setting of BEEP (ON or OFF).
BLOCK	For a marked block, displays the block type, first marked line number, first marked column number, last marked line number, last marked column number, fileid containing the marked block and the string PERSISTANT.
CASE	Displays the case in which text is entered (MIXED or UPPER case), the case setting for string searches (RESPECT or IGNORE case) and the case setting for comparisons made during the CHANGE command (RESPECT or IGNORE case.)
CMDDEF	Displays the current setting of CMDDEF (ALPHA or ALPHANUMERIC) which controls whether numeric values are allowed in macro or command synonym names.
CMDText	Displays the text of the command that would be either executed or placed on the command line had the CMDTEXT command been issued with respect to the current focus column and line.
COLumn	Displays the column number of the focus column.
COLOr COLOur	Displays all the current colour settings, as defined by SET COLOUR, for fields in the focus edit view.
CURSor	Displays the line number of cursor in window, the column number of cursor in window, the line number of the cursor in file (-1 if on the command line) and the column number of cursor in file (-1 if on the command line).
DEFPROFile	Displays the name of default profile macro.
DISPlay	Displays displays the minimum and maximum displayable selection levels.
DSN	Displays the data set name of the current file.
DSORG	Displays the Data Set Organisation of the current file. (PDS, SEQ, KSDS, ESDS, RRDS, LDS, CMS or LIBR)
ENVVars	Displays the current setting for ENVVAR (ON or OFF) and the current environment variable delimiter character.
EOLIn	Displays the current EOL (end-of-line) setting for EOLIN which is used for the GET *fileid* command when *fileid* is an HFS path name.
EOLOut	Displays the current EOL (end-of-line) setting for EOLOUT which is used for save output to an HFS file.

`FLSCReen`	Displays the file line number of the first and last text line visible in the window view.
`FMode`	Displays the file mode of the current file.
`FName`	Displays the file name of the current file.
`FPath`	Displays the file path of the current file.
`FType`	Displays the file type of the current file.
`FIDChanged`	Displays the current setting of the FIDCHANGED flag (ON or OFF).
`FILEId`	Displays the full fileid of the current file.
`HEXSTRING`	Displays the current setting of HEXSTRING (ON or OFF).
`HSCROLLCursor`	Displays the current setting for HSCROLLCURSOR (ON or OFF).
`IMPMACro`	Displays the current setting of IMPMACRO (ON or OFF).
`INIFILE`	Displays the current system and user CBLIINI file.
`INIVARs`	Displays the current system and user CBLIINI file variables and their values.
`INSTANCE`	Displays the current setting of INSTANCE (SINGLE or MULTIPLE).
`INTERFace`	Displays the current setting for INTERFACE (CBLe or ISPF).
`ISPFMODE`	Displays the current setting for ISPFMODE (ON or OFF).
`KEY`	Displays the start and end columns containing the key in the current KSDS file.
`LASTmsg [ASIS]`	Displays the last message or error message generated for the current file view. This message may not heve been displayed if NOMSG was used or MSGMODE was OFF. The ASIS sub-parameter supresses the string "EDT010I LASTMSG" which, by default, is prefixed to the last message string.
`LCOLor` `LCOLour`	Displays the active LCOLOR definitions in the order that they were entered. The output also displays the SET CASE and SET ZONE options that were active when each LCOLOR command was issued.
`LENgth`	Displays the length of data in the focus line. Trailing blanks are not included in this value.
`LIne`	Displays the line number of the current line within the current file.
`LINENd`	Displays the current setting of LINEN (ON or OFF) including the linend character.
`LINEFLAG`	Displays line flag settings for the focus line (NEW or NONEW, CHANGE or NOCHANGE and TAG or NOTAG).
`LISTFILEACTion`	Displays the current setting for LISTFILEACTION (BROWSE, EDIT or NONE).
`LOADWarning`	Displays the current file edit load warning threshold.
`LRECL`	Displays the defined LRECL value for the current file and the longest record encountered when the file was loaded for edit.
`LScreen`	Displays size and location information about the current logical screen (MDI child window and MDI client area). These values are whst would need to be specified on the CBLe SET options, WINSIZE and WINPOS, in order to achieve the size and location display characteristics of the focus edit view. Values are displayed in the following order: 1. MDI child window depth (number of rows). 2. MDI child window width (number of columns). 3. MDI child window vertical position within the client area (row number). 4. MDI child window horizontal position within the client area (column number). 5. MDI client area depth (number of rows). 6. MDI client area width (number of columns).

`MACRO <macroname>`	Displays the name and contents of each macro loaded into storage by a DEFINE command. If <macroname> is specified, then output is restricted to the specified macro. The following message marks the start of the defined macro: `"MACRO Name=<macroname> Text=Loaded from <location> ..."`
`MACROPath`	Displays the current macro path.
`MBR`	Displays the member name of the current file.
`MDILIst`	Displays the current setting for MDILIST (ON or OFF) and whether message lines wrap (WRAP or NOWRAP).
`MSGLine`	Displays the current setting for MSGLINE (ON or OFF), the line number of the first line used for messages, the number of lines that can be used for messages and OVERLAY if the first message line can overlay a file line.
`MSGMode`	Displays the current setting for MSGMODE (ON or OFF).
`NBWindow`	Displays the total number of open edit views, the number of edit views displaying the current file and the current edit view number for the current file.
`OPSys`	Displays operating system name and release.
`PFnn`	Displays the currently assigned definition for the PFKey, PFnn.
`Point`	Displays the focus line number and all names currently allocated to the focus line. If no names are allocated to the focus line (line number n), the following message is returned: EDT010I POINT No points assigned to line n
`Point *`	Displays the line number and associated line names of all named lines in the current file.
`PREfix`	Displays the current setting for PREFIX (ON or OFF) including the relative position of the prefix area in the CBLe window (LEFT or RIGHT) and its width in number of bytes.
`PSCOpe`	Displays the current setting for PSCOPE (ALL or DISPLAY) which determines whether excluded lines are respected or ignored by CBLe prefix or ISPF Edit style line commands.
`RANGE`	Displays the line number of the first and last lines in the current range.
`RECFM`	Displays the current file's Record Format.
`REDO` `REDO *`	Same as UNDO.
`RESERved`	Displays the line numbers of all reserved lines in the current edit view.
`RING`	Displays the number of files edited in the ring of edit views, followed by information on each edited file. (i.e. Alteration counts, number of records and fileid.)
`SCALe`	Displays the current setting for SCALE (ON or OFF) including the location of the scale line in the CBLe window.
`SCOLour` `SCOLor`	Displays the active SCOLOR definitions in the order that they were entered. The output also displays the SET CASE and SET ZONE options that were active when each SCOLOR command was issued.
`SCOPE`	Displays the current setting for SCOPE (DISPLAY or ALL).
`SCReen`	Displays the number of rows and columns available in the 3270 terminal screen.
`SELect`	Displays the selection level of the focus line and the maximum selection level of all lines in the current file.
`SHADow`	Displays the current setting of SHADOW (ON or OFF).

`SIze`	Displays the number of lines in the current file.
`SIZEWarning`	Displays the current file edit file size warning threshold.
`STAY`	Displays the current setting of STAY (ON or OFF).
`STReam`	Displays the current setting of STREAM (ON or OFF).
`SYNonym`	Displays the current setting for SYNONYM (ON or OFF).
`SYNonym *`	Displays all defined synonyms in the order in which they were entered. (i.e. synonym name and definition.)
`THIGHlight`	Displays the current setting for THIGHLIGHT (ON or OFF followed by ALL or FIRST).
`UNDO`	Displays for the current file, the number of change levels that may be undone and redone, and the amount of storage (KB) currently used to store undo information. The format is: `EDT010I UNDO Undos=u  Redos=r Storage=nK`
`UNDO *`	Displays change levels (Edit TRansactions - ETR) and a description of the functions performed for each change level (Edit Transaction Element - ETE) that exist in UNDO storage and may be undone using the UNDO command.
`UNDOING`	Displays the current setting for UNDOING (ON or OFF), the maximum number of change levels and the maximum amount of memory (KB) that will be allocated by CBLe to store UNDO information.
`USERname`	Displays the current user's logon userid.
`VARblank`	Displays the current setting of VARBLANK (ON or OFF).
`VERSION`	Displays the CBLe version: "CBLe" followed by the version number, the build date and build time.
`WINNAMe`	Displays the internal name for current edit view and the frame window (MDI parent window).
`WINPOS`	Displays the respective row, column positions of the current edit view within the client area and the current frame window within the screen display.
`WINSIZe`	Displays the rows x columns size of the current edit view, the current edit view's display area, the MDI parent window, the MDI parent window's display area and the 3270 terminal screen.
`WRap`	Displays the current setting of WRAP (ON or OFF).
`Zone`	Displays the current left zone column and right zone column.

See Also:

EXTRACT
SET

QUEUE

Syntax:

```
>>-- QUEUE --- cmd_stream ---------------------------------------------------><
```

Description:

Primarily for use in CBLe REXX macros, QUEUE stacks a command stream that gets executed the next time the screen is refreshed. This is an effective method of piping commands to a non-CBLe edit view, e.g. List windows.

A screen refresh occurs when the macro execution completes or when the REFRESH command is executed during macro execution.

Any number of command streams may be stacked for subsequent execution in the sequence first in, first out. The command stream is executed from the window view within the CBLe MDI environment that last received focus and may consist of several commands separated by the command line end separator character.

Parameters:

cmd_stream
: Any valid executeable command stream appropriate to the focus window.

Examples:

```
'queue  where recfm=VB ;  sizewindow d=30 w=60'
'listdataset  NBJ.TEST*.**'
'refresh'
```

LISTDATASET opens a new MDI child window within CBLe. When the REFRESH command is executed, the LISTDATASET window (the focus window) receives and executes the queued WHERE and SIZEWINDOW commands before the screen is repainted by CBLi.

QUIT, QQUIT

Syntax:

```
>>----- QUIT ----------------><
```

```
>>----- QQuit ---------------><
```

Description:

QUIT removes the current file from the file ring and from CBLe memory without first saving it to disk. If the file has been changed and not yet saved to disk (i.e. ALT is set to a value >0) then CBLe opens a dialog window asking whether you want to save the changes to the file before it is removed from memory.

QQuit removes the current file from the file ring and from CBLe memory regardless of whether or not changes to the file have been saved to disk.

Parameters:

None.

See Also:

CANCEL
FILE

REDO

Syntax:

```
    >>-- REDO ----------------><
```

Description:

REDO re-applies a change made to the current file that has been undone by a previous UNDO command. Where UNDO has not been issued, executing REDO has no affect.

This command is equivalent to selecting "Redo" from the "Edit" menu on the CBLe window menu bar.

Multiple levels of changes undone by repeated UNDO commands may be re-applied by same number of repeated REDO commands.

Following one or more executions of UNDO, REDO will recover the changes even after the following have occured:

- The cursor position changes.
- The file is saved.
- Changes are made to other files in the file ring.

However, REDO is not able to recover changes following an UNDO if subsequently:

- Further changes are made to the file (lines updated, deleted or added.)
- Changes are made to the selection level of any line (e.g. on an ALL command.)
- Changes are made to the line flags of any line (e.g. on a TAG command.)
- Changes are made to the line name of any line (e.g. on a SET POINT command.)

The third number following "Alt=" on the status line indicates the number of change levels. If this number is followed by an "*" (asterisk), then it is possible to REDO previous UNDO commands.

Parameters:

None.

See Also:

UNDO

REFRESH

Syntax:

```
>>-- REFRESH --------------------------------------------------------------><
```

Description:

For use in a CBLe/SDE REXX macro, the REFRESH command will refresh the edit display during execution of the macro.

The display is not normally updated until a macro is completed or unless keyboard input is required.

Note: Frequent refresh of the display within a macro can increase the macro's run time.

Examples:

`imm do 10; 'add1'; 'refresh'; end`
Add 10 blank lines, one at a time, and refresh the view after each line added.

REPLACE

Syntax:

```
>>-- Replace --+------------+----------><
              |            |
              +-- string --+
```

Description:

REPLACE is also an ISPF primary command.
If INTERFACE=ISPF is in effect (default for CBLi in an MVS environment), then the ISPF version of the command is executed instead. See section ISPF/CBLe CLI Command Precedence and IBM publication "ISPF: Edit and Edit Macros".
Use the CBLe CLI command ECOMMAND to override.

Replace the focus line with the specified text string. The text string begins immediately after the single separating blank that follows the REPLACE command verb.

Parameters:

string Replace text string.
Where more than 1 blank separates the text string from the command verb, the additional separating blanks are treated as part of the text string.

If string is omitted, the focus line is replaced with a blank line.

Examples:

```
r Hello World
```
Focus line is replaced with "Hello World starting in column 1.

```
replace
```
Focus line is replaced with a blank line.

RESET

Syntax:

```
>>-- RESet ---------------><
```

Description:

RESET is also an ISPF primary command.
If INTERFACE=ISPF is in effect (default for CBLi in an MVS environment), then the ISPF version of the command is executed instead. See section ISPF/CBLe CLI Command Precedence and IBM publication "ISPF: Edit and Edit Macros".
Use the CBLe CLI command ECOMMAND to override.

Reset (unmark) the currently marked block.

Parameters:

None.

See Also:

MARK

RESTORE

Syntax:

```
>>-- RESTore ------------><
```

Description:

Restore values of SET options that have been saved by a previous PRESERVE command. RESTORE is most often used in macros.

Parameters:

None.

See Also:

PRESERVE

RIGHT

Syntax:

```
            +- 1 ----+
            |        |
>>-- RIght -+--------+---><
            |        |
            +- n ----+
            |        |
            +- HALF -+
```

Description:

RIGHT is also an ISPF primary command.
If INTERFACE=ISPF is in effect (default for CBLi in an MVS environment), then the ISPF version of the command is executed instead. IBM publication "ISPF: Edit and Edit Macros".
Use the CBLe CLI command ECOMMAND to override.

Position the focus column one or more columns to the right of the current focus column. Alternatively, position the focus column to the right of the current focus column, for a number of columns equal to one half the width of the edit view.

Where the specified number is greater than the number of columns to the right of the focus column (i.e. beyond the LRECL value for the file) then the last column becomes the focus column.

Parameters:

`n`	The number of columns to the right of the current focus column at which the new focus column is to be set. If omitted this parameter defaults to 1.
`HALF`	Position the focus column a number of columns equal to half the width of the edit view, from the current focus column position.

Example:

`right`	Set the focus column to be 1 column to the right of the current focus column.
`ri10`	Set the focus column to be 10 columns to the right of the current focus column.

See Also:

LEFT

SAVE, SSAVE

Syntax:

```
>>-- SAVE ---+------------+-------------------------------------------------><
             |            |
             +-- fileid --+

>>-- SSAVE --+------------+-------------------------------------------------><
             |            |
             +-- fileid --+
```

Description:

Save the current edited file to disk with an associated fileid.

If a *fileid* is specified, the fileid in the title bar of the CBLe edit view is updated and the data is saved under the new fileid. File data that exists on disk under the original fileid is unchanged.
This is equivalent to performing the "Save As" item from the File drop down menu.

If *fileid* is not specified, SAVE attempts to write the file to disk using the currently assigned fileid. By default, this fileid is that which was used to initially edit the file, unless subsequently updated by a SET FILEID, FNAME, FTYPE, FMODE, FPATH command.
This is equivalent to performing the "Save" item from the File drop down menu.

If the fileid to be used by SAVE does not already exist, a new file will be created. For MVS Sequential, PDS(E) and VSAM data sets, this will open the Allocate NonVSAM or Define VSAM dialog window prompting the user to provide the required new data set characteristics.

SAVE will fail and return an error if either of the following conditions are true:

- The fileid used to save the data differs from that used on the initial edit of the file and this fileid already exists for a file on disk.
- For HFS files, the current "Modified" timestamp of the file is later than the time at which the file was last saved, or else first edited, in the current CBLe session. i.e. the file's data has been changed by some other process or user edit.

SSAVE will save the file regardless of the above error conditions.

Where *fileid* is identified as being an HFS path (i.e. begins with "." or contains "/"), DSORG is set to be HFS and the file is saved with the permission mode 740 and the following record delimitation:

- If the current RECFM is F, the HFS file will contain no EOL delimiters and each record will be padded to the current LRECL value.

- For RECFM V or U, HFS file records will be delimited with EOL (end-of-line) characters as defined by the current setting of EOLOUT.

Parameters:

`fileid`
: The fileid to be assigned to the file on writing it to disk.

  For MVS data sets, *fileid* may be the DSN of a Sequential or VSAM data set, the DSN and member name of a PDS(E) library member or an HFS absolute or relative path name.

  For VSE files, *fileid* may be the full LIBR member id, lib.sublib.mname.mtype.

  For CMS files, *fileid* may be the full CMS file id, FNAME FTYPE FMODE (or FNAME.FTYPE.FMODE) If only two qualifiers are specified, FMODE defaults to the current FMODE, and if only one qualifier is specified, then FTYPE and FMODE default to the current FTYPE and FMODE.

See Also:

SET FILEID
FILE

SDATA

Syntax:

```
>>-- SData -- sde_command ---------------------------------------------------><
```

Description:

Direct a command to the CBLi Structured Data Environment (SDE).

The SDATA command allows SDE commands to be issued from a CBLe text-edit window, typically using PF4 to point-and-shoot at commands stored in a command-centre (CMX) file, such as your HOME-file.

Parameters:

sde_command	Any SDE command.

Examples:

```
<sdata create structure   CBL.CBLI.STRUCT(COMPSTR)    \
     from cobol CBL.COPYBOOK.COBOL(COMPDEF)
<sd edit   CBL.SDE.EMP   using CBL.CBLI.STRUCT(COMPSTR)
<sd select Key,InvNumb,DeliveryDate from Orders in CBL.CBLI.STRUCT(COMPSTR)
```

SET

Syntax:

```
>>--+-------+--- option -- value --><
    |       |
    +-- SET -+
```

Description:

Set CBLe environment options.

Parameters:

`option`	The option to be updated.
`value`	The new value for the SET option.

Currently supported SET options are:

ALT ARBCHAR BEEP CASE CMDLINE COLOR COLOUR DEFPROFILE DISPLAY DSN DSORG ENVVARS FMODE FNAME MBR FPATH FTYPE FIDCHANGED FILEID HEXSTRING HSCROLLCURSOR	IMPMACRO INSTANCE ISPFMODE INTERFACE KEY LCOLOR LCOLOUR LINEFLAG LINEND LISTFILEACTION LOADWARNING LRECL MACROPATH MDILIST MSGLINE MSGMODE PFKEY POINT PREFIX PSCOPE RANGE	RECFM RESERVED SCALE SCOLOR SCOLOUR SCOPE SELECT SHADOW SIZEWARNING STAY STREAM SYNONYM THIGHLIGHT UNDOING VARBLANK VIEW WINNAME WINPOS WINSIZE WRAP ZONE

Example:

`SET DISPLAY 10 20`	Display only lines with selection levels between 10 and 20.
`POINT .z`	Assign the name ".z" to the focus line.

See Also:

SET Options
EXTRACT
PRESERVE
QUERY
RESTORE

SETPT

Syntax:

```
              +---- 70 -----+  +--- 132 ---+
              |             |  |           |
>>-- SETPT ---+-------------+--+-----------+--+-----+-----------------------><
              |             |  |           |  |     |
              +- start_col -+  +- end_col -+  +- Q -+
                                              |     |
                                              +- V -+
```

Description:

A single execution of SETPT assigns a label name to multiple lines within a file based on the line data found between the specified column boundary limits. Compare with SET POINT, which unconditionally sets an individual label name on the focus line, or setting a label by simply typing the label name in a line's prefix area.

If a blank delimited word of no more than 9 characters beginning with "." (dot/period) **starts** within the column limits defined by *start_col* and *end_col*, then a label name equal to the word string is assigned to that line. Multiple labels will be assigned to the same line if more than one word matching this criteria is found within the column limits.

Note that a word within the data that extends beyond the *end_col* column limit may still be eligible as a label name so long as the preceding "." is within the column limits.

If a single label is set, then the assigned label name is displayed in the line's prefix area. Multiple labels assigned to the same line are not displayed in the line's prefix area.

The same label name may not be assigned to more than one line in the current file. Therefore, the last occurrence of a label name within the file will be assigned by SETPT. All previous occurrences of the same label name will be unassigned.

Like SET POINT, SETPT takes effect at the File level.

Parameters:

start_col
: Defines the left column limit in a range of columns in which to search for blank delimited, "." prefixed words. Default is column 70.

end_col
: Defines the right column limit in a range of columns in which to search for blank delimited, "." prefixed words. Default is column 132.

`Q`
: Suppress the default report message, "n points have been set by SETPT."

`V`
: List the label names that have been set by SETPT before reporting the total number of points set by SETPT.

Examples:

`SETPT 56 63 V`
: For the following lines of data will set the label ".SMS" on line 238 and label names ".RACF" and ".SEC" on line 239. These label names will be included in the list of label names that have been set on completion of the SETPT command.

```
       |...+....1....+....2....+....3....+....4....+....5....+....6....+...
000238 <edit NBJ.CONFIG.CMX(SMS)    | ** SMS  Configuration ** .SMS
000239 <edit NBJ.CONFIG.CMX(RACF)   | ** RACF Configuration ** .RACF .SEC .X
```

SHIFT

Syntax:

```
                             +- 1 -+  +-------- 1 -------+
                             |     |  |                  |
>>-- SHift --+-- Left ---+--+-----+--+------------------+-->< 
             |           |  |     |  |                  |
             +-- Right --+  +- n -+  +-- group-target --+
```

Description:

Shift text to the LEFT or RIGHT by the specified number of columns. Text is moved on the focus line and on lines up to, but not including, the line containing the first match for group-target.

The SHIFT command affects text from the left zone column or the left boundary of a marked box block, through to the LRECL column. Text that is shifted left beyond the left zone column, the left box block boundary or right beyond the LRECL column, is truncated.

Parameters:

`Left` `Right`	Direction in which to shift text.
`n`	Integer specifying the number of columns by which the text is to be shifted.
`group-target`	Group-target condition defining the end of the target area for the command. If the group-target is not satisfied then the SHIFT command will fail.

Examples:

`shift right`
Starting at the left zone column, text at the focus line is shifted 1 column to the right.

`sh r 6 all`
Starting at the left zone column, text in all lines of the edit view is shifted 6 columns to the right.

`sh L 10 /abc/`
Starting at the left zone column, text at the focus line and in all lines up to, but not including, the first line following the focus line to contain string "abc", is shifted 10 columns to the left.

`sh L 1 block`
Starting at the left boundary of the marked block, text at the focus line and in all lines within the marked block, is shifted 1 column to the left.

SORT

Syntax:

```
>>-- SORT -- group-target --+-----------------------------------------+-->< 
                            |                                         |
                            | +- Ascending (1) -+                     |
                            | |                 |                     |
                            +-+-----------------+--+-- col1 --+--------+
                              |                 |  ^          |        |
                              +- Descending ----+  |          +- col2 -+
                                                   |                   |
                                                   +-------------------+
```

(1) Default for all sort fields until Descending is specified, in which case the default is reversed until Ascending is specified, etc.

Description:

SORT is also an ISPF primary command.
If INTERFACE=ISPF is in effect (default for CBLi in an MVS environment), then the ISPF version of the command is executed instead. See section ISPF/CBLe CLI Command Precedence and IBM publication "ISPF: Edit and Edit Macros".
Use the CBLe CLI command ECOMMAND to override.

Sort lines in the target area specified by group-target in ascending or descending order.

Characters between the specified start and end columns are compared against those characters in equivalent columns on all lines in the target area.

The SORT command processes all lines in the target area, regardless of line selection level.

Were STAY is set ON, the focus line is unchanged by the SORT command. Otherwise, the line that becomes the first line in the target area following the SORT command, also becomes the focus line.

Parameters:

`group-target`	Group-target condition defining the end of the source target area. If the group-target is not satisfied then the SORT command will fail.
`ASCENDING`	Sort the field that follows in ascending or descending order. ASCENDING is default.
`DESCENDING`	The specified sort order prevails for all sort fields that follow until otherwise specified.
`col1 col2..`	Pair of column numbers defining the leftmost and rightmost columns of a sort field. Any number of sort field column pairs may be specified. Precedence is given to those sort fields specified first in the list. "*" (asterisk) may be specified for col2 indicating that the current right zone column should be used. Where col2 is not specified for the last sort field column pair, then "*" (asterisk) is implied. Where col1 and col2 are not specified, the current ZONE columns are used. If, however, the target area is a box block, the left and right boundaries of the block are used.

Example:

`sort 10 10 20`	Sort the focus line and the following 9 lines in ascending order based on the single sort field specified as columns 10 to 20.
`sort :20`	Sort the focus line and all lines up to, but not including, line 20 in ascending order based on the implied sort field defined by the right and left ZONE columns.
`sort /abc/ d 55 58 10 12 a 3 8`	Sort the focus line and all lines up to, but not including, the next line below the focus line that contains the string "abc". These lines are sorted firstly in descending order based on sort field specified as columns 55 to 58, in descending order based on sort field specified as columns 50 to 12, and finally in ascending order based on sort field specified as columns 3 to 8.

SOS

Syntax:

```
>>-- SOS --- action -->< 
```

Description:

Perform Screen Operation Simulation to action cursor placement and text editing. in CBLe macros.

Parameters:

`action` Perform the explicit SOS action.

Currently supported SOS actions are:

`ADDline` `LINEAdd`	Add a blank line following the focus line. The cursor is positioned in column 1 of the new line.
`DELLine` `LINEDel`	Delete the focus line. The line following becomes the focus line.
`MAKECURR`	If the cursor in not on the command line, make the cursor line the current line.
`REFResh`	Refresh the 3270 display. Use SOS REFResh in macros which invoke external commands which might alter the 3270 display in ways which are not noticed by CBLi (such as ISPF commands). The next time CBLi displays the 3270 screen it will rebuild it completely thus clearing any screen formatting done by the external function.

See Also:

ADD
DELETE
REFRESH

SPLIT

Syntax:

```
>>-- SPlit --+-------------+--><
             |             |
             +-- ALigned --+
```

Description:

Split the focus line into two lines starting at the focus column. Text at, and to the right of, the focus column is moved to column 1 of a new line following the focus line.

The focus line pointer and focus column pointer are unchanged following SPLIT.

Parameters:

`ALIGNED` A number of leading blanks, equal to the number of leading blanks in the focus line, are inserted before the text split onto the new line.

Example:

In the following, focus column is column 21.

```
Command>

<...+....1....+....2|...+....3....+....4....+
      Hello Jane,   Goodbye John

Command> split

<...+....1....+....2|...+....3....+....4....+
      Hello Jane,
Goodbye John



Command> split al

<...+....1....+....2|...+....3....+....4....+
      Hello Jane,
      Goodbye John
```

See Also:

JOIN
SPLTJOIN

SPLTJOIN, SJOIN

Syntax:

```
>>--+-- SPLTJOIN --+---><
    |              |
    +-- SJoin -----+
```

Description:

Perform a SPLIT ALIGNED or JOIN ALIGNED on the focus line.

If there exists text at, or to the right of, the focus column in the focus line, then SPLIT ALIGNED is executed, otherwise JOIN ALIGNED is executed.

Parameters:

None.

See Also:

JOIN
SPLTJOIN

STEMINSERT

Syntax:

```
>>-- STEMInsert --- rexx_stemvar -------------------------------------><
```

Description:

For use in CBLe REXX macros, STEMINSERT is a record mass insert command based on a REXX compound (stem) variable.

STEMINSERT determines the number of lines to insert from the *rexx_stemvar*.0 value and inserts new lines with text obtained from the value *rexx_stemvar*.***n_line***, where ***n_line*** is the insert line index (***n_line***=1,2,3,...,*rexx_stemvar*.0). The new lines are inserted following the focus line.
STEMINSERT is a fast method of insert and should be used in place of the following REXX syntax:

```
do  i =  1  to  linetext.0
  'insert' linetext.i
end
```

Replace this with:

```
'steminsert'  linetext
```

Parameters:

rexx_stemvar
: The stem portion of an assigned REXX compound variable.

SUBMIT

Syntax:

```
>>-- SUBmit --+------------+------><
              |            |
              +-- fileid --+
```

Description:

Submit the specified batch job file to the MVS or VSE batch system. The file should contain a valid Job Card and Job Control.

If fileid is not specified, the file in the current CBLe edit view is submitted.

The EDT032I job submitted message is returned containing the job name and job number.

`EDT032I Job xxxxxxxx(JOBnnnnn) submitted.`

The job name and job number may subsequently be used in the CBLi command EO to edit (read only) the job's output listing.

Parameters:

`fileid`	Full fileid of the file to be submitted to batch.

Example:

`submit`	Submit the currently edited job to batch.
`submit cbl.jcl(scanpds)`	Submit MVS job CBL.JCL(SCANPDS) to batch.

SYNEX

Syntax:

```
>>-- SYNEX -- command -------><
```

The SYNEX (SYNonym EXecute) command is for use in CBLe REXX macros.

When SYNONYM ON is in effect, a command entered from a CBLe command line is automatically checked to determine whether it is the name of a defined synonym. If so, the action taken will be that specified by the synonym definition.

By default, CBLe macros do not perform any synonym processing. Prefixing the command with SYNEX when SYNONYM ON is in effect, will force CBLe to check whether the command to be executed has had its behaviour defined via a SET SYNONYM command. If not, the command is executed as normal.

Parameters:

`command`	Any synonym name, CBLe command or macro name followed by its parameters.

Examples:

`synex CC red`
Check existance of "CC" as a synonym name. Failing that, treat it as a CBLe command or macro name.

See Also:

COMMAND
SET SYNONYM

SYSCOMMAND, TSO, CMS, DOS

Syntax:

```
>>--+- SYScommand -+--- command --->< 
    |              |
    +- TSO --------+
    |              |
    +- CMS --------+
    |              |
    +- DOS --------+
```

Description:

Pass the command directly to the local CMS or TSO environment for execution.

When a command is issued to CBLe, the following occurs:

1. If the command is recognised as a CBLe command it is executed by CBLe.
2. If the command is not recognised as a CBLe command and IMPMACRO is set ON, then CBLe checks for a matching macro name and, if found, executes the macro.
3. If the command is not a macro name or IMPMACRO is set OFF, CBLe passes the command and control to the CMS or TSO environment.

Parameters:

`command`	Valid CMS or TSO command or expression.

Example:

`cms query dasd`	Pass the command "query dasd" to CMS.

See Also:

CBLI
MACRO

SYSEDIT

Syntax:

```
>>-- SYSEdit ---+------------+----------------------------------------------><
                |            |
                +-- fileid --+
```

Description:

For CMS or ISPF environments only, open the system text editor, XEDIT or ISPF Edit, to edit the specified file.

SYSEDIT does not support HFS files.

System edit of the file in the current CBLe edit view may also be achieved by selecting **System Edit** from the **Edit** menu item in the CBLe Main Menu Bar.

If the file is already open in a CBLe edit view and has alterations, SYSEDIT fails with the following error message:
`EDT150E You must save the changes to this file before using SYSEDIT.`

Having made changes to the file in the system editor, saving the changes and quiting from that editor will refresh any existing CBLe edit views displaying the file. The following edit warning message is returned:
`EDT151W This file has been refreshed because you made changes to it in SYSEDIT.`

Parameters:

fileid
: The full fileid of the file to be edited.
Default is the current fileid.

See Also:

EDIT

TAG

Syntax:

```
>>-- TAG --+---------------+--><
           |               |
           +- line-target -+
```

Description:

Tag (set the tag bit on) and highlight all lines matching the specified line-target.

Where line-target is not specified, highlighting is removed and the tag bit is set off for all lines in the file.

Parameters:

`line-target`	Line-target condition.

Example:

`tag /#*/`	Highlight and tag all lines containing the string "#*".

TFIND

Syntax:

```
>>-- TFind --+-----------------+--><
             |                 |
             +- string-target -+
```

Description:

Locate the first line from the focus line to contain the specified line-target in the 1st position of the current ZONE (BOUNDS) and make this line the new focus line.

If WRAP is set ON and the line-target is not found before bottom-of-file or top-of-file is encountered, then the scan continues from the opposite extreme of the file up to and including the focus line.

Where string-target is not specified, CBLe repeats the last TFIND command issued.

Parameters:

`string-target`	Line-target condition.

Example:

`TF "//} DD"`
Find next DD card in MVS JCL, where 'ARBchar ON }' and default 'ZONE 1 *' are in effect.

`TFIND ~/   /`
Find next line not beginning with 3 blanks.

`TF /++/ | /###/`
Find next line beginning with either 2 plus or 3 hash signs.

See Also:

LOCATE
SET ARBCHAR
SET WRAP
SET ZONE

TOP

Syntax:

```
>>-- TOP --->< 
```

Description:

Make the Top of File line the focus line.

Parameters:

None.

See Also:

BOTTOM

UNDO

Syntax:

```
>>-- UNDO ----><
```

Description:

Undo one level of changes made to the current file.

This command is equivalent to selecting "Undo" from the "Edit" menu on the CBLe window menu bar.

Multiple levels of changes may be undone by repeated UNDO commands.

The third number following "Alt=" on the status line indicates the number of change levels. If this number is not zero, then changes may be undone using UNDO.

Parameters:

None.

See Also:

REDO

UP

Syntax:

```
             +- 1 -+
             |     |
>>-- Up ---+-----+---><
             |     |
             +- n -+
```

Description:

UP is also an ISPF primary command.
If INTERFACE=ISPF is in effect (default for CBLi in an MVS environment), then the ISPF version of the command is executed instead. See section ISPF/CBLe CLI Precedence and IBM publication "ISPF: Edit and Edit Macros".
Use the CBLe CLI command ECOMMAND to override.

Position the focus line one or more lines above the current focus line.

Where the specified number is greater than the number of lines above the focus line, the Top of File line becomes the focus line.

Parameters:

`n` The number of lines above the current focus line at which the new focus line is to be set. If omitted this parameter defaults to 1.

Example:

`up`	Set the focus line to be 1 line above the current focus line.
`u10`	Set the focus line to be 10 lines above the current focus line.

See Also:

DOWN

UPPERCASE

Syntax:

```
    >>-- UPPercase -----+----------------+--><
                        |                |
                        +- group-target -+
```

Description:

Upper case all alpha characters in the target area.

Only alpha characters that are within the current ZONE columns will be upper cased.

Where BLOCK is specified as the group-target and the target area is a box block, then the ZONE setting is ignored and all alpha characters within the block are upper cased.

The focus line is not changed by a UPPERCASE command.

Parameters:

`group-target` Group-target condition defining the end of the source target area. If the group-target is not satisfied then the UPPERCASE command will fail.

Example:

`uppercase 6`	Alpha characters in the focus line and the 5 lines following are upper cased.
`upp -8`	Alpha characters in the focus line and the 7 lines preceding are upper cased.
`upper /SELC/`	Alpha characters in the focus line and all lines up to, but not including, the first line following the focus line to contain the string "SELC", are upper cased.

See Also:

LOWERCASE

VIEW

Syntax:

```
>>-+- View ---+--+---------------------------------------------------------+--><
   |          |  |                                                         |
   +- Browse -+  +- fileid --+--------------------------------------------+
    (1)                      |                                            |
                             +- ( -+- PROFILE macroname -+-+--------------+
                                   |                     | |              |
                                   +- NOPROFile ---------+ +-| HFS Opts |-+
```

HFS Opts:

```
        +- EOL ---+------------+---------------+
        |         |            |               |
        |         +-- CR ------+               |
        |         |            |               |
        |         +-- LF ------+               |
        |         |            |               |
        |         +-- NL ------+               |
        |         |            |               |
        |         +-- CRLF ----+               |
        |         |            |               |
        |         +-- LFCR ----+               |
        |         |            |               |
        |         +-- CRNL ----+               |
        |         |            |               |
        |         +-- string --+               |  +- LRECL lrecl -+
        |                                      |  |               |
 >------+--------------------------------------+--+---------------+----->
        |                                      |
        +- RECFM ---- F -----------------------+
```

Notes:

1. BROWSE is a synonym of VIEW for VSE and CMS systems only.

Description:

VIEW is equivalent to EDIT except that data is edited read-only. See EDIT for a full details on supported parameters.

Attempting to save alterations (SAVE, FILE) to data edited as read-only will fail with a message that suggests changing the fileid and retrying the operation. See SET FILEID, or one of its associated SET options, for changing the edited data's fileid.

Changing the fileid to that of a file that already exists and then executing a save will return an error. Use SSAVE or FFILE is overwrite the existing file data.

VIGNORE

Syntax:

```
>>-- VIGnore ---- command ---------------------------------------------------><
```

Description:

When ENVVARS is set ON, VIGNORE may be used as a prefix to a command string to temporarily set ENVVARS OFF for the duration of the command execution.

Note that if the command executed is a CBLe REXX macro, then variable translation is suspended for all commands within the macro unless they are prefixed with VRESPECT or ENVVARS is explicitly set back ON.

If ENVVARS is set OFF, the VIGNORE prefix has no effect.

Parameters:

`command`	Any command followed by its parameters.

Example:

`VIgnore   Change   /%user%/%system%/ 1 *`
All occurances of the literal string "%user%" are changed to the string "%system" on the focus line.

`VIgnore   EQU    LOADLIB   %PREFIX%.LOAD`
Macro EQU sets user environment variable "LOADLIB" to the literal string "%PREFIX%.LOAD".

`VRespect   LL    %LOADLIB%`
List library members belonging to "%PREFIX%.LOAD" where "%PREFIX" may be set to different values for separate invocations of this LL command.

See Also:

EDITV
VRESPECT
SET ENVVARS

VRESPECT

Syntax:

```
>>-- VRespect -- command ------->&lt;
```

Description:

When ENVVARS is set OFF, VRESPECT may be used as a prefix to a command string to temporarily set ENVVARS ON for the duration of the command execution.

Note that if the command executed is a CBLe REXX macro, then variable translation will be obeyed for all commands within the macro unless they are prefixed with VIGNORE or ENVVARS is explicitly set back OFF.

If ENVVARS is set ON, the VRESPECT prefix has no effect.

Parameters:

`command`	Any command followed by its parameters.

Example:

```
VRespect  Change   /%user%/%system%/ 1 *
```

Both variables "user" and "system" are substituted and all occurances on the focus line of the value substituted for "user" are changed to the the value substituted for "system".

```
VRespect  EQU    LOADLIB   %PREFIX%.LOAD
```

The current value of variable "prefix" is replaces "%PREFIX%" in "%PREFIX%.LOAD" and macro EQU sets user environment variable "LOADLIB" to the resolved string.

See Also:

EDITV
VIGNORE
SET ENVVARS

WINDOW

Syntax:

```
>>-- Window --+----- NEXTwindow ---------------------+----------><
              |                                      |
              +----- PREVwindow ---------------------+
              |                                      |
              +----- CAScade ------------------------+
              |                                      |
              |                   +-- HOr -------+   |
              |                   |              |   |
              +----- TILE --------+--------------+---+
              |                   |              |   |
              |                   +-- Vert ------+   |
              |                                      |
              +----- ARRANGEicons -------------------+
              |                                      |
              +----- NEWwindow ----------------------+
              |                                      |
              +----- HEX ----------------------------+
              |                                      |
              |                   +-- DOCument --+   |
              |                   |              |   |
              +----- CLOse -------+--------------+---+
              |                   |              |   |
              |                   +-- FRAme -----+   |
              |                   |              |   |
              |                   +-- FILE ------+   |
              |                                      |
              +----- MENUmode ----+--------------+---+
              |                   |              |   |
              |                   +-- DOCument --+   |
              |                   |              |   |
              |                   +-- FRAme -----+   |
              |                   |              |   |
              |                   +-- FILE ------+   |
              |                   |              |   |
              |                   +-- Edit ------+   |
              |                   |              |   |
              |                   +-- ACTions ---+   |
              |                   |              |   |
              |                   +-- OPTions ---+   |
              |                   |              |   |
              |                   +-- WINdow ----+   |
              |                   |              |   |
              |                   +-- Help ------+   |
              |                                      |
              |                   +-- DOCument --+   |
              |                   |              |   |
              +--+-- RESTore ---+--+--------------+---+
                 |              |  |              |
                 +-- MINimise --+  +-- FRAme -----+
                 +-- MINimize --+
                 |              |
                 +-- MAXimise --+
                 +-- MAXimize --+
```

Description:

Perform window focusing, positioning and sizing operations on the current edit (document) view or MDI parent (frame) window.

Parameters:

`NEXTWINDOW`
Place focus on the next edit view in the ring.

`PREVWINDOW`
Place focus on the previous edit view in the ring.

`CASCADE`
Cascade all MDI child windows within the MDI parent window.

`TILE`
Tile all MDI child windows within the MDI parent window.

HOR	horizontally (default.)
VERT	vertically.

`ARRANGEICONS`
Arrange all minimised MDI child windows so that they are lined up along the bottom of the MDI parent display area.

`NEWWINDOW`
Open an new edit view for the data in the current edit view.

`HEX`
Open a hex view of the focus line.
A hex view is a storage display window that contains a hexadecimal and character representation of the line of edited data. The contents of the line may be updated by the user in either of the data representations, simply by overtyping the existing data and hitting <Enter>.

PF7 and PF8 scroll up and down respectively through the displayed data, whereas PF10 and PF11 scroll backwards and forwards through the lines of edited data.

`CLOSE`
Close the specified window type.

DOCUMENT	Current document window (edit view) (default.)
FRAME	Frame window (MDI parent) and all its child windows.
FILE	All edit views that contain the file currently being edited.

`MENUMODE`
Places the cursor at the specified menu bar item and, where possible, opens the drop down menu. If no item is specified the cursor is simply placed at the first item of the menu bar.

DOCUMENT	Document window (edit view) system menu.
FRAME	Frame window (MDI parent) system menu.
FILE	File drop down menu.
EDIT	Edit drop down menu.
ACTIONS	Actions drop down menu.
OPTIONS	Options drop down menu.
WINDOW	Window drop down menu.
HELP	Help item.

`MAXIMISE`
`MAXIMIZE`
Maximise the current edit view (DOCUMENT), the default, or the MDI parent (FRAME) window.

`MINIMISE`
`MINIMIZE`
Minimise the current edit view (DOCUMENT), the default, or the MDI parent (FRAME) window.

`RESTORE`
Restore the current edit view (DOCUMENT), the default, or the MDI parent (FRAME) window back to its original state, prior to being maximised or minimised.

Examples:

`win max`
Maximise the current edit view.

`win new`
Open a new edit view for the data in the current edit view.

`win clo file`
Close all edit views containing the file currently being edited.

See Also:

SET WINPOS
SET WINNAME
SET WINSIZE

WINDOWCOMMAND

Syntax:

```
>>-+- WINDOWCOMMAND -+-- command --->< 
   |                 |
   +- WINCMD --------+
```

Description:

Execute a CBLi windows management command.

The WINDOWCOMMAND command has been introduced to enable window manipluation commands to be issued from CBLe REXX macros.

Examples of CBLi windows commands are MOVEWINDOW, SIZEWINDOW and the SQL style SELECT. See the supplied macro LDX for an example of its use

Parameters:

`command`
: Any command followed by its parameters.

Example:

```
WinCmd mw x=3 y=26
WinCmd szw w=72 d=31
WinCmd select entry,org,lrecl,recfm,blksz,referenced,vol,pdse,created
```

See Also:

CBLI
SYSCOMMAND

SET Options

The SET command allows the user to specify options that tailor the CBLe working environment.

SET options may be initialised via the following methods:

1. The CBLe section of the CBLI INI file which is processed at CBLi startup.
2. The PROFILE macro which is executed at the start of every CBLe session.
3. The CBLe command line, a CBLe macro or the "SET Options" entry of the "Options" menu on the CBLe window menu bar. Command dialog box.

SET options take effect at the following different levels:

Global	The option affects all edited files.
File	The option may be set differently for each file.
View	The option may be set differently for each CBLe window view of the same file.

Supported SET options:

ALT ARBCHAR BEEP CASE CMDLINE COLOR COLOUR DEFPROFILE DISPLAY DSN DSORG ENVVARS FMODE FNAME MBR FPATH	FTYPE FIDCHANGED FILEID HEXSTRING HSCROLLCURSOR IMPMACRO INSTANCE ISPFMODE INTERFACE KEY LCOLOR LCOLOUR LINEFLAG LINEND LISTFILEACTION LOADWARNING	LRECL MACROPATH MDILIST MSGLINE MSGMODE PFKEY POINT PREFIX PSCOPE RANGE RECFM RESERVED SCALE SCOLOR SCOLOUR SCOPE	SELECT SHADOW SIZEWARNING STAY STREAM SYNONYM THIGHLIGHT UNDOING VARBLANK VIEW WINNAME WINPOS WINSIZE WRAP ZONE

SET ALT

Syntax:

```
>>-- SET -- ALT -- n1 --+------+-------------------------------------------><
                        |      |
                        +- n2 -+
```

Description:

Update the CBLe or SDE edit window alteration count which is reported on the status line.

SET ALT takes effect at the File level.

"Alt=n1,n2;x" on the status line indicates the number of alterations made to the file since it was last saved or autosaved, the number of alterations made to the file since it was last saved (regardless of intervening autosaves) and the number of change levels that may be undone using UNDO. The "*" (asterisk) indicates that change levels may be re-applied with REDO.

Note: autosave is not currently supported.

Parameters:

n1
: Number of changes since last save or autosave.

n2
: Number of changes since last save.

Example:

`set alt 10`
: Set alteration count since last SAVE or AUTOSAVE to 10.

See Also:

EXTRACT
QUERY

SET ARBCHAR

Syntax:

```
>>--+-------+- ARBchar --+- ON --+--+------+--+------+--><
    |       |            |       |  |      |  |      |
    +- SET -+            +- OFF -+  +- c1 -+  +- c2 -+
```

Description:

Activate and optionally allocate arbitrary characters (ARBCHAR characters). ARBCHAR characters are treated as wildcard characters when used in strings for line-target, column-target and the CHANGE command.

SET ARBCHAR takes effect at the View level.

Parameters:

`c1` ARBCHAR character representing zero or more characters in a string. Default is "$" (dollar).

`c2` ARBCHAR character representing a single character in a string. Default is "?" (Question Mark).

Example:

`arbch on % #` Activate ARBCHARs "%" (percent) and "#" (hash). The command CLOCATE /1%2#3/ would be successful for:

```
12a3
1112a3
12222b3
133334442h3
```

However, it would fail for:

```
123
1112ab3
```

`arbch off` Deactivate use of ARBCHARs.

See Also:

EXTRACT
QUERY

SET BEEP

Syntax:

```
>>--+-------+- BEEP ---+- ON --+---><
    |       |          |       |
    +- SET -+          +- OFF -+
```

Description:

Determines whether the speaker beeps when an error message is returned.

SET BEEP takes effect at the Global level.

Example:

`beep on`	Beep on all subsequent error messages.

See Also:

EXTRACT
QUERY

SET CASE

Syntax:

```
>>--+-------+- CASE --+- Mixed -+--+-----------+--+-----------+--><
    |       |         |         |  |           |  |           |
    +- SET -+         +- Upper -+  +- Respect -+  +- Respect -+
                                   |           |  |           |
                                   +- Ignore --+  +- Ignore --+
```

Description:

Determine the way in which the case of character strings are interpreted.

SET CASE takes effect at the View level.

Parameters:

`MIXED` `UPPER`	For all new text typed in the CBLe window view, alpha characters appear in upper or lower case (MIXED), as controlled via the user 's keyboard, or entirely in upper case (UPPER). Default is MIXED.
`RESPECT` `IGNORE`	For string target searches and SORT fields, either IGNORE or RESPECT the case of each alpha character in the search string or sort field. Default is IGNORE.
`RESPECT` `IGNORE`	For the CHANGE command, either IGNORE or RESPECT the case of each alpha character in string1. i.e. with IGNORE in effect, all strings that match string1, regardless of capitalisation, will be changed to string2. Default is RESPECT.

Example:

`case u i i`	New text appears in upper case, case is ignored for string searches and for the CHANGE command string1 parameter.

1. INSERT abcABC would input text as "ABCABC".
2. /abc/ would locate the next line containing one of "abc", "Abc", "ABc", "ABC", "aBc", "aBC" or "abC".
3. CHANGE/abc/def/* * would change all occurrences of "abc", "Abc", "ABc", "ABC", "aBc", "aBC" and "abC" to "def".

See Also:

EXTRACT
QUERY
CHANGE
SORT

SET CMDDEF

Syntax:

```
>>--+-------+- CMDDEF ---+- ALPHA --------+----------------------------------><
    |       |            |                |
    +- SET -+            +- ALPHANUMERIC -+
```

Description:

Determines whether numeric values are allowed in macro or command synonym names.

By default, the CBLe interface allows specification of numeric parameters or relative line targets to be adjoined to the preceding or following command verb with no intervening blanks (i.e. CMDDEF ALPHA). e.g. "3add6" will locate the 3rd line following the focus

line and then add 6 new lines.

This means that, by default, numerics contained within macro names and command synonyms are interpreted by the command processor and not considered part of the command verb. (This may be bypassed for macro execution by preceding the macro name with the MACRO command.)

If CMDDEF ALPHANUMERIC is in effect (default for the CBLe ISPF interface), then numerics within command verbs, macro names and synonyms are not interpreted by the command processor. e.g. "3add6" will attempt execute a command synonym or macro with name "3add6".

SET CMDDEF takes effect at the Global level.

Parameters:

`APLHA`
`APLHANUMERIC`
: Specifies whether command verbs contain alphanumeric characters or only alpha characters.
Default for INTERFACE=ISPF is ALPHANUMERIC. Default for INTERFACE=CBLE is ALPHA.

See Also:

EXTRACT
QUERY

SET CMDLINE

Syntax:

```
>>--+-------+- CMDline --+----------+----><
    |       |            |          |
    +- SET -+            +- TOP ----+
                         |          |
                         +- BOTtom -+
```

Description:

Position the command line at the top or bottom of the display window.

If no parameter is specified, the Command Line window is opened. The Command Line window may also be opened via the System Menu.

SET CMDLINE takes effect at the View level.

Parameters:

`TOP` `BOTTOM`	Location of command line. TOP positions the command line at the first line following the title bar while BOTTOM positions the command line immediately following the horizontal scroll bar. Default is BOTTOM.

Example:

`cmdline top`	Position the command line at the top of the display window.

See Also:

EXTRACT
QUERY

SET COLOR, SET COLOUR

Syntax:

```
                                                              +- NONe ------+
                                                              |             |
>>--+-------+--+- COLOr --+--+- Arrow ------+--+- Blue ------+--+-------------+--><
    |       |  |          |  +- PROmpt -----+  |             |  |             |
    +- SET -+  +- COLOur -+  |              |  +- Red -------+  +- BLInk -----+
                             +- Block ------+  |             |  |             |
                             |              |  +- Pink ------+  +- REVvideo --+
                             +- COMmand ----+  |             |  |             |
                             +- Cmdline ----+  +- Green -----+  +- Uscore ----+
                             |              |  |             |
                             +- FCURsor ----+  +- Turquoise -+
                             |              |  |             |
                             +- Filearea ---+  +- Yellow ----+
                             |              |  |             |
                             +- HIDE -------+  +- White -----+
                             |              |  |             |
                             +- HIghlight --+  +- Default ---+
                             |              |
                             +- Msgline ----+
                             |              |
                             +- Pending ----+
                             |              |
                             +- PRefix -----+
                             |              |
                             +- Scale ------+
                             |              |
                             +- SHadow -----+
                             |              |
                             +- THIghlight -+
                             |              |
                             +- TOfeof -----+
```

Description:

Set colours for fields in the CBLe window.

SET COLOUR takes effect at the File level.

Parameters:

`ARROW` `PROMPT`	Command prompt "Command>". Default colour is WHITE.
`BLOCK`	Text within a marked block field. Default colour is BLUE REVVIDEO.
`COMMAND` `CMDLINE`	Command line text. Default colour is RED.
`FCURSOR`	The Find Cursor within File area, set by the FIND or Change dialog windows. Default colour is WHITE REVVIDEO.
`FILEAREA`	Untagged, unmarked text within File area. Default colour is BLUE
`HIDE`	Excluded lines hidden by ISPF edit HIDE command. Default colour is WHITE USCORE.
`HIGHLIGHT`	Highlighted (tagged) lines. Default colour is YELLOW.
`MSGLINE`	Message lines (with MSGLINE ON). Default colour is RED.
`PENDING`	Pending commands in the prefix area. Default colour is RED.
`PREFIX`	Prefix area (with PREFIX ON). Default colour is GREEN.
`SCALE`	Scale line (with SCALE ON). Default colour is WHITE.
`SHADOW`	Shadow lines (for excluded lines when SHADOW ON). Default colour is WHITE.
`THIGHLIGHT`	Highlighted targets. Default colour is GREEN.
`TOFEOF`	Top of File and End of File lines. Default colour is YELLOW.
`BLUE` `RED`	Supported colours on each field. If DEFAULT is specified, the default colour for the field is set.

```
PINK
GREEN
TURQUOISE
YELLOW
WHITE
DEFAULT
```

`BLINK` `REVVIDEO` `USCORE` `NONE` — Extended highlighting of the specified field. The colour may blink, be displayed in reverse video or be underlined. Default is NONE.

Examples:

```
set colour prompt green uscore
```

```
set colour hi g bli
```
Text in tagged lines will be green and blinking.

```
col com turq rev
```

See Also:

EXTRACT
QUERY
SET LCOLOUR
SET SCOLOUR

SET DEFPROFILE

Syntax:

```
>>--+-------+--- DEFPROFile --- macroname -------->< 
    |       |
    +- SET -+
```

Description:

Define the name of the default profile macro that gets executed every time a new file is added to the ring of edited files.

SET DEFPROFILE takes effect at the Global level.

Parameters:

`macroname` — Name of a CBLe REXX macro.
This may be the full data set name and member name or simply the name of a member found in the macropath libraries.

Examples:

```
defprof cbleprof
```

```
defprof cbl.test.cble(profile)
```

See Also:

EXTRACT
QUERY

SET DISPLAY

Syntax:

```
>>--+-------+- DISPlay -- n1 --+------+--><
    |       |                  |      |
    +- SET -+                  +- n2 -+
```

Description:

Display only lines with the specified selection level or lines whose selection level falls within the specified range.

SET DISPLAY takes effect at the View level.

Every line in the edited file is initially assigned a selection level of 0. The selection level of a line may be changed to any number in the range 0 to 255 via the SET SELECT command. The selection level of a line may automatically be changed by the ALL command and when using the X or S prefix area command.

Initially, DISPLAY 0 0 is set and so all lines in the file are displayed.

Parameters:

`n1`
: The selection level of the lines to be displayed or the low value in a range of selection levels for which lines will be displayed.

`n2`
: The high value in a range of selection levels for which lines will be displayed. "*" (asterisk) may be specified to indicate 255. If not specified, n2 is equal to n1.

Example:

`DISP 1`	Display lines with a selection level of 1.
`DISPLAY 0 1`	Display all lines with a selection level between 0 and 1.
`DISPLAY 10 20`	Display all lines with a selection level between 10 and 20.

See Also:

EXTRACT
QUERY
ALL
SET SELECT

SET DSN

Syntax:

```
>>--+-------+-- DSN --- dsname --------------------------------------------><
    |       |
    +- SET -+
```

Description:

For HFS paths only, DSN is equivalent to FILEID.

Change the DSN (Data Set Name) portion of the fileid assigned to the currently edited in-storage data. The fileid in the title bar of the window view is updated to reflect the change and the FIDCHANGED flag is set on. Following a save operation, the edited data will be saved to disk using the changed fileid.

SET DSN takes effect at the File level.

Parameters:

dsname
: For MVS data sets, *dsname* replaces the full data set name (excluding any PDS member name) of the current file and must contain at least two qualifiers. For HFS paths, *dsname* is the full fileid of the HFS file.

  For VSE files, dsname replaces the full lib.sublib.mname.mtype LIBR member id of the current file. i.e. the full fileid.

  For CMS files, dsname replaces the full FNAME FTYPE FMODE of the current file. Dsname is specified as FNAME.FTYPE.FMODE where each qualifier must be a valid CMS fileid token. If only two qualifiers are specified, then only the FNAME and FTYPE are replaced. Similarly, if only one qualifier is specified, then only the FNAME is replaced.

Example:

`DSN CBL.VVC.LST.V210`
: Change the MVS PDS name of the current file.

`DSN PRD2.CBLINST.VNF210.TXT`
: Change the VSE LIBR member id of the current file.

`DSN PROFILE.CBLe`
: Change the CMS FNAME and FTYPE of the current file. **Note:** FMODE is unchanged.

See Also:

EXTRACT
QUERY
SET FILEID
SET FNAME
SET FTYPE
SET FMODE
SET FPATH

SET DSORG

Syntax:

```
>>--+-------+--- DSORG ---+- PDS ------------------+------------------------><
    |       |             |                        |
    +- SET -+             +- SEQ ------------------+
                          |                        |
                          +- KSDS -- kp1 --- kp2 --+
                          |                        |
                          +- ESDS -----------------+
                          |                        |
                          +- RRDS -----------------+
```

Description:

Change the data set organisation for the currently edited data. DSORG may only be set for data sets that have not yet been allocated or defined.

SET DSORG takes effect at the File level.

When changing the DSORG to PDS, the FNAME (penultimate) qualifier is removed from the DSN and used as the member name. If only 2 qualifiers exist in the original DSN, then the command will fail.

Coversely, when changing DSORG from PDS to any other supported organisation, the member name is inserted into the DSN as the penultimate qualifier.

Note: SET FNAME is synonymous with SET MBR.

Parameters:

`PDS`
`SEQ`
`KSDS`
`ESDS`
`RRDS`
: Supported data set organisations.

kp1
kp2
: Start and end positions of the key field. These are required if defining the data to be a VSAM KSDS data sets.

Examples:

`dsorg ksds 1 16`
: Make the currently edited data a VSAM KSDS with KEY at position 1 (offset 0) for length 16.

`dsorg pds`
: Make the currently edited data a PDS library member.

See Also:

EXTRACT
QUERY
SET FNAME
SET KEY

SET ENVVARS

Syntax:

```
>>--+-------+- ENVVars -+- ON --+--+------+--><
    |       |           |       |  |      |
    +- SET -+           +- OFF -+  +- c1 -+
```

Description:

ENVVARS (Environment Variables) determines whether translation of CBLe environment variables occurs in commands issued from CBLe edit view windows. This includes commands executed from the command prompt or from within an edited file (typically a CMX file) via the CMDTEXT facility.

ENVVARS also defines the character used to delimit variable names.

CBLe environment variables include standard system determined variables, MVS System symbolics, EDITV variables and CBLiINI vaiables.

SET ENVVARS takes effect at the View level.

Parameters:

`ON` `OFF`	ON and OFF switches translation on and off respecively. If ENVVARS is OFF, command string will be passed without translating variables. Default is ON.
`c1`	The delimiter character. If c1 is not specified, the delimiter character last defined in the current CBLe view, is unchanged. Default is "%" (percent - X'6C'.)

Example:

```
envv on  #
```
Set the variable delimiter character to "#" (hash).

See Also:

EDITV
VIGNORE
VRESPECT

SET EOLIN

Syntax:

```
>>--+-------+-- EOLIn ------+------------+-----------------------------------><
    |       |               |            |
    +- SET -+               +--- CR -----+
                            |            |
                            +--- LF -----+
                            |            |
                            +--- NL -----+
                            |            |
                            +--- CRLF ----+
                            |            |
                            +--- LFCR ----+
                            |            |
                            +--- CRNL ----+
                            |            |
                            +--- string --+
```

Description:

EOLIN alters the current input EOL (end-of-line) delimiter string used to interpret variable length records obtained from an HFS file via the GET command.

An EOLIN value is set for all SDE and CBLe edit views including those containing non-HFS files. In CBLe edit views, this allows use of the GET and ISPF style COPY commands to retrieve records from an HFS file into a Sequential file, VSAM file or PDS(E) member.

When an edit view is opened and before the edit data is read, the default EOLIN is automatically set to be one of the following values, in the order of precedence:

1. The EOL parameter argument specified on the EDIT or BROWSE command.

2. For SDE EDIT/BROWSE only, the EOLIN value set in the SDE profile macro (using SET EOLIN).
3. The EOL format value defined in the directory entry.
4. EOLIN=NL (new line).

SET EOLIN takes effect at the File level.

Parameters:

`CR|LF|NL|CRLF|LFCR|CRNL|`*`string`*
Identifies the end-of-line delimiter. Delimiter elements are as follow:

NL	X'15'	New Line.
CR	X'0D'	Carriage Return.
LF	X'0A'	Line Feed.
string	-	A 2-byte user specified character or hex string.

See Also:

SET EOLOUT
GET

SET EOLOUT

Syntax:

```
>>--+-------+-- EOLOut -----+------------+----------------------------------><
    |       |               |            |
    +- SET -+               +-- CR ------+
                            |            |
                            +-- LF ------+
                            |            |
                            +-- NL ------+
                            |            |
                            +-- CRLF ----+
                            |            |
                            +-- LFCR ----+
                            |            |
                            +-- CRNL ----+
                            |            |
                            +-- string --+
```

Description:

EOLOUT determines the output HFS file EOL (end-of-line) delimiter string to be used when saving edited data to an HFS fileid which is not fixed format. i.e RECFM F is **not** the current setting for the edited data.

By default, the EOLOUT value is equal to the EOLIN value established when the CBLe edit view is initially opened for edit or browse.

An EOLOUT value is also set for non-HFS files allowing the user to subsequently save the data in the edit view to a new HFS file simply by using the SAVE *fileid* command where fileid is an HFS path name.

SET EOLOUT takes effect at the File level.

Parameters:

`CR|LF|NL|CRLF|LFCR|CRNL|`*`string`*
Identifies the end-of-line delimiter. Delimiter elements are as follow:

NL	X'15'	New Line.
CR	X'0D'	Carriage Return.
LF	X'0A'	Line Feed.
string	-	A 2-byte user specified character or hex string.

See Also:

SET EOLIN
SAVE

SET FMODE

Syntax:

```
>>--+-------+--- FMode ---- qual ---------------------------------------------><
    |       |
    +- SET -+
```

Description:

Change the FMODE (File Mode) portion of the fileid assigned to the currently edited in-storage data. The fileid in the title bar of the window view is updated to reflect the change and the FIDCHANGED flag is set on. Following a save operation, the edited data will be saved to disk using the changed fileid.

SET FMODE takes effect at the File level.

Parameters:

qual
: For MVS data sets, *qual* replaces the high level qualifier of the current data set name. For HFS files, *qual* sets the absolute HFS path name's first level directory name above the root directory for the currently edited data. (e.g. "usr" in "/usr/include/arpa/inet.h")

  For VSE files, *qual* replaces the lib LIBR library name.

  For CMS files, *qual* replaces the FMODE of the current file.

Example:

`FM CBL110`
: Change the MVS High Level qualifier of the current file.

`FM PRD1`
: Change the VSE LIBR library name of the current file.

`FM G1`
: Change the CMS FMODE of the current file.

See Also:

EXTRACT
QUERY
SET DSN
SET FILEID
SET FNAME
SET FTYPE
SET FPATH

SET FNAME, SET MBR

Syntax:

```
>>--+-------+--+- FName -+-- qual -------------------------------------------><
    |       |  |         |
    +- SET -+  +- MBR ---+
```

Description:

Change the Member/File Name portion of the fileid assigned to the currently edited in-storage data. The fileid in the title bar of the window view is updated to reflect the change and the FIDCHANGED flag is set on. Following a save operation, the edited data will be saved to disk using the changed fileid.

SET FNAME and SET MBR take effect at the File level.

Parameters:

`qual`
: For MVS sequential data sets, *qual* replaces the penultimate qualifier of the current data set name. For MVS PDS data sets, *qual* replaces the member name. For HFS files, *qual* sets the file name portion of the HFS path name for the currently edited data. (e.g. "inet" in "/usr/include/arpa/inet.h")

  For VSE files, *qual* replaces the mname LIBR member name.

  For CMS files, *qual* replaces the FNAME of the current file.

Example:

`FN TEST001`
: Change the MVS PDS member name.

`FN V210INST`
: Change the VSE LIBR member name of the current file.

`FN CBLVJ27`
: Change the CMS FNAME of the current file.

See Also:

EXTRACT
QUERY
SET DSN
SET FILEID
SET FMODE
SET FTYPE
SET FPATH

SET FPATH

Syntax:

```
>>--+-------+--- FPath ---- qual -------------------------------------------><
    |       |
    +- SET -+
```

Description:

Change the FPATH (File Path) portion of the fileid assigned to the currently edited in-storage data. The fileid in the title bar of the window view is updated to reflect the change and the FIDCHANGED flag is set on. Following a save operation, the edited data will be saved to disk using the changed fileid.

SET FPATH takes effect at the File level.

Parameters:

`qual`
: For MVS data sets, *qual* replaces all qualifiers except the low level qualifier of the current data set name. For HFS files, *qual* sets the file path portion of the HFS path name for the currently edited data. (e.g. "/usr/include/arpa" in "/usr/include/arpa/inet.h")
  Changing the file path from an MVS DSN to an HFS file path will change the DSORG to HFS.

For VSE files, *qual* replaces the lib.sublib LIBR library and sublibrary names.

For CMS files, *qual* replaces the FMODE of the current file.

Example:

`FP CBL.CMD`
Change all but the low level qualifier of the current MVS data set.

`FP PRD2.CBL`
Change the VSE LIBR lib.sublib name of the current file.

`FP A1`
Change the CMS FMODE of the current file.

See Also:

EXTRACT
QUERY
SET DSN
SET FILEID
SET FMODE
SET FNAME
SET FTYPE

SET FTYPE

Syntax:

```
>>--+-------+--- FType ---- qual ------------------------------------------><
    |       |
    +- SET -+
```

Description:

Change the FTYPE (File Type) portion of the fileid assigned to the currently edited in-storage data. The fileid in the title bar of the window view is updated to reflect the change and the FIDCHANGED flag is set on. Following a save operation, the edited data will be saved to disk using the changed fileid.

SET FTYPE takes effect at the File level.

Parameters:

qual
For MVS data sets, *qual* replaces the low level qualifier of the current data set name. For HFS files, *qual* sets the file type portion of the HFS path name for the currently edited data. (e.g. "h" in "/usr/include/arpa/inet.h")

For VSE files, *qual* replaces the LIBR member type.

For CMS files, *qual* replaces the FTYPE of the current file.

Example:

`FT S210`
Change the low level qualifier of the current MVS data set.

`FT OBJ`
Change the VSE LIBR member type of the current file.

`FT EXEC`
Change the CMS FTYPE of the current file.

See Also:

EXTRACT
QUERY
SET DSN
SET FILEID
SET FMODE
SET FNAME
SET FPATH

SET FIDCHANGED

Syntax:

```
>>-+-----------+-- FIDCHanged ----+-- ON ----+----------------------------><
   |           |                  |          |
   +- SET -----+                  +-- OFF ---+
```

Description:

This option allows the user to manually control the internal CBLe flag that indicates that the fileid of the current file has been changed by the user. (i.e. via the SET options DSN, FMODE, FNAME, MBR, FPATH, FTYPE or FILEID.)

If the FIDCHANGED flag is on, then CBLe QUIT or END commands will prompt the user to save the file before the window is closed.

The SET FIDCHANGED functionality allows the user to override CBLe's axiom that any change to the fileid assigned to the data edited in storage will prompt for a save operation. e.g. Consider the following steps:

1. Assign a temporary fileid (e.g. via SET DSN) to the data being edited, so releasing the exclusive MVS SPFEDIT ENQ or VSE LIBR LOCK on the original DSN. This automatically sets the FIDCHANGED flag on.
2. Perform operations requiring an exclusive ENQ or LOCK on the original fileid. (e.g. DELETE)
3. Restore the original fileid to the edited data.
4. Set FIDCHANGED off so that, when executing QUIT or END, the user is not prompted to save the file.

This is the technique employed in the distributed ERA and RENAME CBLe REXX macros.

SET FIDCHANGED takes effect at the File level.

Parameters:

`ON|OFF`
    FIDCHANGED flag is set ON or OFF.

See Also:

EXTRACT
QUERY
SET DSN
SET FILEID
SET FMODE
SET FNAME
SET FPATH
SET FTYPE

SET FILEID

Syntax:

```
>>--+-------+- FIleid -- fileid ---------------------------------------------><
        |       |
        +- SET -+
```

Description:

Change the full fileid assigned to the currently edited in-storage data. The fileid in the title bar of the window view is updated to reflect the change and the FIDCHANGED flag is set on. Following a save operation, the edited data will be saved to disk using the changed fileid.

SET Fileid takes effect at the File level.

Parameters:

`fileid`
: For MVS data sets, *fileid* replaces the full DSN of the current data set name. For PDSs, this includes the member name. For HFS path names, *fileid* sets the absolute or relative fileid for the currently edited data. Changing the fileid from an MVS DSN to an HFS file will change the DSORG to HFS.

For VSE files, *fileid* replaces the full lib.sublib.mname.mtype LIBR member id of the current file.

For CMS files, *fileid* replaces the full FNAME FTYPE FMODE of the current file. *fileid* is specified as FNAME.FTYPE.FMODE where each qualifier must be a valid CMS fileid token. If only two qualifiers are specified, then only the FNAME and FTYPE are replaced. Similarly, if only one qualifier is specified, then only the FNAME is replaced.

Example:

`FI  CBL.VVC.LST.V210(TEST003)`
: Change the current fileid to an MVS PDS name and member id.

`FILE  /u/cbl/nbj/oem.cbl.selcnam`
: Change the current fileid to an HFS path name.

`FI  PRD2.CBLINST.VNF210.TXT`
: Change the full VSE LIBR member id of the current file.

`FI  PROFILE.CBLe`
: Change the CMS FNAME and FTYPE of the current file. **Note:** FMODE is unchanged.

See Also:

EXTRACT
QUERY
SET DSN
SET FMODE
SET FNAME
SET FPATH
SET FTYPE

SET HEXSTRING

Syntax:

```
>>--+-------+- HEXstring ---+- ON --+---><
    |       |               |       |
    +- SET -+               +- OFF -+
```

Description:

HEX is also an ISPF primary command.
If INTERFACE=ISPF is in effect (default for CBLi in an MVS environment), then the ISPF version of the command is executed instead. See section ISPF/CBLe CLI Command Precedence and IBM publication "ISPF: Edit and Edit Macros".
Use the CBLe CLI command ECOMMAND to override.

Controls interpretation of hexadecmal strings.

A valid hexadecimal string consists of 1 or more pairs of hexadecimal digits (0-9, A-F), enclosed in "'" (apostrophes) or '"' (quotes) and immediately preceded by an upper or lower case "X". A pair of hexadecimal digits represent a single value in the range x'00' to x'FF' each corresponding to a single character. e.g. The following are equivalent:

```
'ABCDE'
x'C1C2C3C4C5'
```

SET HEXSTRING takes effect at the View level.

Parameters:

ON
OFF

With HEXSTRING ON, CBLe will interpret any string argument enclosed in "'" (apostrophes) or '"' (quotes) and immediately preceded by an upper or lower case "X", as being hexadecimal. With HEXSTRING OFF, strings of this type are not interpreted.
Default is OFF.

See Also:

CHANGE
Targets

SET HSCROLLCURSOR

Syntax:

```
>>--+-------+--- HSCROLLCursor ---+-- ON ---+------><
    |       |                     |         |
    +- SET -+                     +-- OFF --+
```

Description:

Controls horizontal scrolling when the target of a successful CLOCATE command is outside the displayed width of the edit view.

When HSCROLLCURSOR is in effect and the target column is outside the edit view, the view is scrolled horizontally so that the target (new focus) column becomes the first column displayed in the edit view.

SET HSCROLLCURSOR takes effect at the View level.

Parameters:

ON OFF	Set HSCROLLCURSOR ON or OFF. Default is OFF.

Examples:

```
hscrollc on
```

See Also:

EXTRACT
QUERY
CLOCATE

SET IMPMACRO

Syntax:

```
>>--+-------+- IMPMACro --+- ON --+--><
    |       |             |       |
    +- SET -+             +- OFF -+
```

Description:

IMPMACRO (Implied Macro) determines whether CBLe will search for a macro when a command is issued that does not match a CBLe command name. If IMPMACRO is ON and a macro exists with the same name as the command issued, then the macro is executed.

If a matching macro name is not found or IMPMACRO is OFF, then an error message is issued.

SET IMPMACRO takes effect at the Global level.

Parameters:

ON OFF	Set IMPMACRO ON or OFF. Default is ON.

See Also:

EXTRACT
QUERY
MACRO

SET INIVAR

Syntax:

```
>>-- SET --------- INIVar ----- var_name --------+-------------+--------------------><
                                                 |             |
                                                 +- var_value -+
```

Description:

SET INIVAR adds or updates a USER.CBLiINI variable in storage. This may be one of CBLi's standard CBLiINI variables or a user defined variable.

Note that most, but not all, standard CBLi variables are only interrogated when CBLi is started. Therefore, a change to these types of variables' values will not take effect until CBLi is re-started.

When the CBLi session is closed, the new or updated CBLiINI variable is written to the USER.CBLiINI data set and so the variable is set across CBLi sessions. New variables are added to the bottom of the CBLiINI data set following a comment line reporting the number of new variables and their date of insert, whereas existing variables are updated in their original location in the file.

The value of any CBLiINI file variable may be referenced directly within CBLe CLI command syntax by simply specifying the variable name between "%" (percent) characters. Note that the level of CBLiINI variable ("USER" or "SYSTEM") is the first qualifier of the variable. e.g. `LD %USER.DEVELOPMENT.PREFIX%`

SET INIVAR takes effect at the Global level.

Parameters:

var_name
: The two-level CBLiINI variable name in the format **nnnnn.mmmmm** to be added or replaced. e.g. DEVELOPMENT.TESTHLQ
The first level represents a sub-section in the CBLiINI file which is inserted in enclosing "( )" (parentheses). e.g. (DEVELOPMENT)
The second level is an entry within the sub-section in the CBLiINI file. e.g. TESTHLQ=*var_value*
The name of a new variable is automatically upper cased prior to being written to the CBLiINI data set. This does not affect references to the variable name as they are not case sensitive. e.g. %user.Development.TestHLQ%

var_value
: The value to be assigned to the CBLiINI variable name.
All tokens following *var_name* are parsed, as though being read from the CBLiINI file, and the resultant value assigned to *var_name*. If *var_value* contains an unquoted "*" (asterisk), it is treated as the start of comment data and so the end of *var_value*. The "*" and any text following is ignored and not written to the CBLiINI file. The significant *var_value* text is passed, unaltered, as the value assigned to *var_name*.
Default is a null string.

Example:

`set inivar Edit.UseDSNPrefix YES`
: Update the value of the existing CBLi standard CBLiINI variable USER.Edit.UseDSNPrefix.

`set inivar Prefix.SysVol Z19`
: Add a user CBLiINI variable for system volume prefix. e.g. To list all system volumes (Z19*):
`LVOL %USER.PREFIX.SYSVOL%*`

`set inivar %user%.LogDSN %user%.LOG.D%yy%%mm%%dd%`
: Add a user CBLiINI variable for a user defined log data set. Both the variable name and its value include standard environment variables which are translated before being passed to the SET INIVAR operation.

SET INSTANCE

Syntax:

```
>>--+-------+- INSTANCE --+- SINGLE ---+--><
    |       |             |            |
    +- SET -+             +- MULTIPLE -+
```

Description:

INSTANCE determines whether multiple instances or only a single instance of CBLe may be active at any one time.

SET INSTANCE takes effect at the Global level.

Parameters:

`SINGLE`
`MULTIPLE`
: If a copy of CBLe is already loaded and a file is edited via a List window prefix command or a menu bar item, then one of the following occurs:

1. For INSTANCE SINGLE, the file will be edited in the existing CBLe window.
2. For INSTANCE MULTIPLE, the file will be edited in a new CBLe window.

See Also:

EXTRACT
QUERY

SET ISPFMODE

Syntax:

```
>>--+-------+--- ISPFMode ---+-- ON ---+-------><
    |       |                |         |
    +- SET -+                +-- OFF --+
```

Description:

When running in an ISPF environment, SET ISPFMODE controls whether 3270 screen I/O is managed by ISPF (ON) or CBL3270 (OFF). This is the same as executing the CBLi command ISPF with no parameters which toggles ISPF screen management on and off.

If SET ISPFMODE is issued in a non-ISPF environment, the following error message is returned:

```
EDT095E ISPF is not available in the current environment.
```

With ISPFMODE ON, the menu bar item **Swap** is added to the CBLi Main Menu which, if selected, will execute **ISPF SWAP** to display an ISPF split screen.

SET ISPFMODE takes effect at the Global level.

Parameters:

`ON`
`OFF`
: Set ISPFMODE ON or OFF.
In an ISPF environment, the default is ON. Otherwise ISPF is OFF.

See Also:

EXTRACT
QUERY

SET INTERFACE

Syntax:

```
                                                 +--VIEW---+
                                                 |         |
>>--+-------+-- INTERFace --+--CBLe--+--+---------+--+--------------+-><
    |       |               |        |  |         |  |              |
    +- SET -+               +- ISPf -+  +- FILE --+  +- INItialise -+
                                                 |         |
                                                 +- GLObal-+
```

Description:

Sets the CBLe file editor interface to either CBLE or ISPF mode.

The CBLE interface is compatible with IBM's CMS editor XEDIT and Mansfield Software's PC adaptation KEDIT.

The ISPF interface is compatible with IBM's TSO ISPF editor and includes support for the most commonly used primary and line (prefix) commands, scrolling and PF key/command line concatenation.

INTERFACE=ISPF is the default on MVS type systems.
INTERFACE=CBLe is the default for VM/CMS and VSE.

The interface can be set globally, at the file or view level, allowing different interfaces for different files and even different views of the same file.

The edit interface can also be set in the SYSTEM and/or USER CBLiINI files, using:

```
 (Edit)
    Interface=ISPF       * ISPF or CBLe.
```

Parameters:

CBLE
ISPF
: Set the current edit interface to CBLE (XEDIT/KEDIT style) or ISPF.

VIEW
FILE
GLOBAL
: Set for the level at which the SET INTERFACE command will take effect.

VIEW	The current edit view only.
FILE	All edit views of the current file.
GLOBAL	All current and any new edit views.

INITIALISE
: Initialise the default display area fields that correspond to the particular edit environment. This uses system defined defaults and and any CBLiINI option overrides. e.g. For ISPF, displays a SCROLL field in the command line. For CBLe, displays a reserved scale line as the first line of the edit display area.

Example:

`INTERFace CBLe`
Set the CBLe edit mode for the current file edit view.

`INTERFACE ISPF FILE INI`
Set the ISPF edit mode for all edit views of the current file and initialise their edit display areas to defaults associated with the ISPF interface.

See Also:

ECOMMAND
ICOMMAND

SET KEY

Syntax:

```
>>--+-------+--- KEY --- kp1 -- kp2 ----><
    |       |
    +- SET -+
```

Description:

Change the key position and key length for the current KSDS file. KEY may only be set for KSDS data sets that have not yet been defined.

SET KEY takes effect at the File level.

Parameters:

`kp1` `kp2`	Start and end positions of the key field. Both are mandatory for SET KEY.

Examples:

```
key 101 116
```

See Also:

EXTRACT
QUERY
SET DSORG

SET LCOLOR, SET LCOLOUR

Syntax:

```
                                              +- Green -----+ +- NONe -----+
                                              |             | |            |
>>--+-------+--+-- LCOLour -+-+- line-target -+-------------+-+------------+-+----------+-+-->&lt;
    |       |  |            | |               |             | |            | |          | |
    +- SET -+  +-- LCOLor --+ |               +- Blue ------+ +- BLInk ----+ +-- name --+ |
                              |               |             | |            |              |
                              |               +- Default ---+ +- REVvideo -+              |
                              |               |             | |            |              |
                              |               +- Pink ------+ +- Uscore ---+              |
                              |               |             |                             |
                              |               +- Red -------+                             |
                              |               |             |                             |
                              |               +- Turquoise -+                             |
                              |               |             |                             |
                              |               +- White -----+                             |
                              |               |             |                             |
                              |               +- Yellow ----+                             |
                              |                                                           |
                              +---------------------+-- name --+-- OFF -------------------+
                                                    |          |
                                                    +--- * ----+
```

Description:

Assign and unassign colours to text on all lines that contain the specified line-target string. The SET LCOLOUR command is influenced by the current setting for CASE and ZONE.

All SET LCOLOUR commands that have been issued for the file, are stored and applied in order to any new text entered in the file.

SET LCOLOUR takes effect at the File level.

Parameters:

`line-target`	Colour only those lines which satisfy the line-target condition. The line-target must be a string target.

`BLUE`
`DEFAULT`
`GREEN`
`PINK`
`RED`
`TURQUOISE`
`WHITE`
`YELLOW`
	Supported colours. If DEFAULT is specified the 3270 hardware default is used.

`BLINK`
`NONE`
`REVVIDEO`
`USCORE`
	Extended highlighting. The colour may blink, be displayed in reverse video or be underlined.
	Default is NONE.

`name`	Unique name by which the SET LCOLOUR entry is referenced. Name may be enclosed within delimiter characters which are included in QUERY and EXTRACT LCOLOUR output but do not form part of the name itself.

Name may be used to reference a specific SET LCOLOUR entry to set it off.

If the name used already exists for a prior SET LCOLOUR command, then the original SET LCOLOUR entry is set off and replaced by the new definition.

If name is not specified, it defaults to being the line-target string.

Note: name does not include any special keywords used in the line-target (i.e. WORD, PREFIX or SUFFIX.)

`*`	(Asterisk) references all SET LCOLOUR entries.
`OFF`	Remove a SET LCOLOUR entry and undo any colouring caused by that entry.

Examples:

```
set lcolour /==/ green uscore equal2

lcol /alloc/ yel rev "tso allocs"

lcol word level blue blink
```

See Also:

EXTRACT
QUERY
SET CASE
SET COLOUR
SET SCOLOUR
SET ZONE

SET LINEFLAG

Syntax:

```
                                +---------------+   +------ 1 -------+
                                v               |   |                |
>>--+-------+-- LINEFLAG --+-- CHAnge ----+---+----------------+--->< 
    |       |              |              |   |                |
    +- SET -+              +-- NOCHAnge --+   +- group-target -+
                           |              |
                           +-- NEW -------+
                           |              |
                           +-- NONEW -----+
                           |              |
                           +-- TAG -------+
                           |              |
                           +-- NOTAG -----+
```

Description:

Each line of a file in an edit view has three associated flag bits, namely the CHANGE bit, NEW bit and the TAG bit.

By default, when CBLe loads a file for edit, all three flag bits are set off. The flag bits are set automatically by CBLe as follow:

CHANGE	Set ON when a line is changed, added, moved or sorted.
NEW	Set ON when a line is added.
TAG	Set on by TAG and MORE TAG and set off by LESS TAG.

SET LINEFLAG allows the user to manually set the flag bits on and off, for lines contained in the specified target area.

SET LINEFLAG takes effect at the File level.

Parameters:

`CHANGE` `NOCHANGE`	Set the change flag bit on/off. Default is NOCHANGE.
`NEW` `NONEW`	Set the new flag bit on/off. Default is NONEW.
`TAG` `NOTAG`	Set the tag flag bit on/off. Default is NOTAG.
`group-target`	

Group-target condition defining the end of the source target area. If the group-target is not satisfied then the DELETE command will fail.

Examples:

`set lineflag nonew`
Set the new flag bit off for the focus line only.

`set lineflag new nochange /##/`
Set the new flag bit on and the change flag bit off for the focus line and all lines up to, but not including, the first line to contain the string "##".

`set lineflag tag 20`
Set the tag flag bit on for the focus line and the 19 lines that follow.

See Also:

EXTRACT
QUERY
TAG

SET LINEND

Syntax:

```
>>--+-------+- LINENd --+- ON --+--+------+-->< 
    |       |           |       |  |      |
    +- SET -+           +- OFF -+  +- c1 -+
```

Description:

LINEND (Line End) determines whether multiple commands may be issued from the CBLe command line by separating each command with the line end character. LINEND may also be used to define the line end character.

CBLe command line by separating each command with the line end character. LINEND may also be used to define the line end character.

SET LINEND takes effect at the View level.

Parameters:

`ON`
`OFF` ON allows and OFF disallows multiple commands on the CBLe command line. OFF If LINEND is OFF, the line end character will be treated as part of a single command stream. Default is ON.

`c1` The line end character. If c1 is not specified, the line end character, last defined in the current CBLe view, is unchanged. Default is "!" (exclamation mark).

Example:

`linen on #` Allow multiple commands on the CBLe command line separated by "#" (hash).

See Also:

EXTRACT
QUERY

SET LISTFILEACTION

Syntax:

```
>>--+-------+-- LISTFILEACTion --+------------+------------><
    |       |                    |            |
    +- SET -+                    +-- Browse --+
                                 |            |
                                 +-- Edit ----+
                                 |            |
                                 +-- NONE ----+
```

Description:

Sets the default action when <Enter> is hit on an entry in a List window.

SET LISTFILEACTION overrides the default set in the SYSTEM and/or USER CBLiINI file by the (System) ListFileAction option.

Although a CBLe command, SET LISTFILEACTION not only affects list windows opened as child windows of the current CBLe session, but also list windows opened from anywhere within CBLi.

SET LISTFILEACTION takes effect at the Global level.

Parameters:

`Browse`	If the entry is a CMS fileid, VSE LIBR member name, MVS DSN of a sequential or VSAM data set, or MVS PDS(E) member name, then hitting <Enter> will edit the entry Read-Only.
`Edit`	If the entry is a CMS fileid, VSE LIBR member name, MVS DSN of a sequential or VSAM data set, or MVS PDS(E) member name, then hitting <Enter> will edit the entry Read-Write.
`NONE`	No action will be taken on hitting <Enter> on any list window entry.

Example:

```
listfileact browse
```
Change default action to Edit read/only.

SET LOADWARNING

Syntax:

```
>>--+-------+-- LOADWarning --+---------+--+----------+--><
    |       |                 |         |  |          |
    +- SET -+                 +-- OFF --+  +-- nnn  --+
                                           |          |
                                           +-- nnnK --+
                                           |          |
                                           +-- nnnM --+
```

Description:

When a file is opened for edit, the contents of the file are first loaded into storage.

LOADWARNING defines the warning threshold value for the current number of bytes loaded for the edited file. When the count of the number of bytes loaded exceeds a value that is a multiple of the loadwarning value, a warning popup message window is displayed.

The purpose of the file load warning threshold is to stop a user from accidentally editing a very large file.

CBLe SET LOADWARNING overrides the Edit.LoadWarning value set by the SYSTEM or USER CBLIINI file. If a LoadWarning value has not been set in either CBLiINI file, the default value is 1M (one megabyte).

SET LOADWARNING takes effect at the Global level.

Parameters:

`ON` `OFF`	Enable or Disable file size checking during load. Default is ON.

`nnn`
`nnnK`
`nnnM`
	Number of bytes loaded after which a warning is issued. This value may be specified as a number of bytes (nnn), number of kilobytes (nnnK) or a number of megabytes (nnnM).

Example:

```
loadw  on  8M
```
When a new file is edited, open a warning popup window after every 8 megabytes of data is loaded.

See Also:

SET SIZEWARNING
EXTRACT
QUERY

SET LRECL

Syntax:

```
>>--+-------+--- LRecl --- n_bytes -----------------------------------------><
    |       |
    +- SET -+
```

Description:

Change the logical record length value of the currently edited in-storage data.

The **RecL=** value in the status bar and LRECL value in the title bar of the CBLe or SDE edit window view are updated to reflect the change.

For SDE edit window views, SET LRECL fails and returns an error if the LRECL value specified is less than the maximum potential record length that may be mapped by any record type that is currently associated with records in the file.

e.g. A record type "Statistics" may contain variable length and variable numbers of fields so that, potentially, a record with a length in the range 125 to 350 may be mapped by this record type. If, when editing a structured file, record type "Statistics" is used to map at least one record, then the LRECL may not be set to a value lower than 350. If further record types are used that are capable of mapping longer records, then this LRECL minimum value will be higher.

Following a save operation, the edited data will be saved to disk using the changed lrecl value. For MVS, LRECL may only be changed where the fileid of the currently edited data is not that of an existing data set. (i.e. a new data set will be allocated when the data is saved.)

SET LRECL takes effect at the File level.

Parameters:

n_bytes

The logical record length.

For a RECFM F (fixed record format) file, *n_bytes* is the length of each record. CBLe will pad short lines with blanks and truncate long lines.

For a RECFM V (variable record format) file, *n_bytes* is the maximum record length. Short lines will not be padded, but longe lines will be truncated.

"*" (asterisk) may be specified to set LRECL equal to the WIDTH value specified when the file was edited. However, by default, the WIDTH value is equal to the LRECL.

See Also:

EXTRACT
QUERY

SET MACROPATH

Syntax:

```
                                +--------------+
                                |              |
                                v              |
>>--+-------+- MACROPath --+-- macrolib --+--><
    |       |
    +- SET -+
```

Description:

MACROPATH defines the list of libraries in which CBLe REXX macros are to be found. Each library is searched in the order specified until the invoked macroname is found.

CBLe SET MACROPATH **replaces** the in storage Edit.MacroPath variable set by the SYSTEM or USER CBLIINI file.

In storage macros, macros issued via the IMMEDIATE command and macros issued using their full fileid via the MACRO command are not affected by MACROPATH.

SET MACROPATH takes effect at the Global level.

Parameters:

`macrolib` A single token (containing no blanks, commas or semi-colons) representing one element (library) of the macro path.

Up to 15 elements may be specified with blank, comma or semi-colon used as the delimiter between each element.

The meaning and format of these elements vary depending on the operating system.

For MVS, each element of the macro path is a PDS/PDSE library. Each PDS/PDSE must be allocated at the time the macro path is defined and must be of equal record format (RECFM). If RECFM=F, then the LRECL must also be the same. The libraries are dynamically concatenated and opened when the SET MACROPATH command is executed.

For VSE, each element of the macro path is a LIBR sub-library and/or a LIBR chain (LIBDEF) name. The sub-library must already exist, but chain names are not checked.

LIBR members are searched for with the following member types (in this order):

1. CBLE
2. CBLEDIT
3. XEDIT

For CMS, each element of the macro path is a file type and optionally a file mode. If specified, the file mode must be separated from the file type with a "." (period). Default file mode is "*" (asterisk) indicating any file mode.

Only files with the given file types and modes will be loadable as macros.

Example:

`SET MACROPATH  USER.CBLEDIT.MACROS  SYSTEM.CBLEDIT.MACROS`	Set the MVS macro path. Note that PDS/PDSE libraries must have the same RECFM.
`SET MACROP  LIB.USER;LIB1.SYS;LIB2.SYS;PROC`	Set the VSE macro path. The last element specifies the LIBDEF PROC search path.
`SET MACROP   CBLETEST.A,CBLE.*`	Set the CMS macro path. All files on disk A with file type CBLETEST then files on all accessed disks with file type CBLE.

See Also:

MACRO
IMMEDIATE

SET MDILIST

Syntax:

```
>>--+-------+-- MDILIst --+-- ON ---+---------->< 
    |       |             |         |
    +- SET -+             +-- OFF --+
```

Description:

Controls whether a CBLi List window, opened from an MDI child window, is itself an MDI child window of the current MDI parent window, or opened as a child window of the CBLi main window.

With MDILIST ON, any new list window is under the control of the MDI application.

Although a list window can be a child window of CBLe, it is not editable. To generate a list as an edit view or assign list entries to REXX stem variables, then use the CBLe command LIST.

SET MDILIST takes effect at the Global level.

Parameters:

`ON` `OFF`	Default is ON.

Examples:

```
mdilist on
```

See Also:

CBLe List, DB2 SQL & CBLVCAT Document Windows
MACRO
IMMEDIATE
LIST

SET MSGLINE

Syntax:

```
>>--+-------+- MSGLine -- ON -- line --+------------------+--><
    |       |                          |                  |
    +- SET -+                          +- n --+-----------+
                                              |           |
                                              +- OVERLAY -+
```

Description:

MSGLINE defines the line number and number of lines CBLe uses for its output message lines. The line number is specified as being relative to the top, middle or bottom of the document window.

SET MSGLINE takes effect at the View level.

Parameters:

`line` Line number of first message line relative to the top, middle or bottom of the document window.

Line numbers relative to the top of the document window are specified as positive integers. e.g. 2, 8

Line numbers relative to the middle of the document window are specified as offsets from M. e.g. M+1, M-3

Line numbers relative to the bottom of the document window are specified as negative integers. e.g. -6, -10

Initially line is set to 1.

`n` The number of message lines. CBLe messages may need to occupy more than one message line. If the number of message lines are exceeded or a message overlaps the command line, a pop-up message window entitled "CBL Text Edit Messages" is opened and all the CBLe messages displayed.

If n is not specified, the number of message lines last defined in the current CBLe view, is unchanged. Initially n is set to 5.

`OVERLAY` Specifies that the first message line should overlay a line normally used to display a line of the file. If omitted, the first message line will be reserved for message display only.

Initially OVERLAY is set on.

Example:

`msgline on 5 20`	Allocate 20 message lines beginning 5 lines down from the top of document window.
`msgline on M+3`	2 Allocate 2 message lines beginning 3 lines down from the middle of document window.
`msgline on -6 6`	Allocate 6 message lines beginning 6 lines up from the bottom of document window.

See Also:

EXTRACT
QUERY

SET MSGMODE

Syntax:

```
                                    +- WRAP ---+
                                    |          |
>>--+-------+- MSGMode --+- ON --+--+----------+----------------------------><
    |       |            |       |  |          |
    +- SET -+            +- OFF -+  +- NOWRAP -+
```

Description:

MSGMODE defines whether messages and error messages are displayed.

Parameters WRAP/NOWRAP are not supported for SDE window views.

SET MSGMODE takes effect at the View level.

Parameters:

`ON`
`OFF`
: MSGMODE ON will allow display of messages and error messages at the defined message lines.

Note: QUERY and EXTRACT LASTMSG will retrieve the text of last message issued regardless of the current setting for MSGMODE.

`WRAP`
`NOWRAP`
: Not supported in SDE window views.
Determines whether message text with length greater than the width of the edit view window, wraps onto the next message line.
Default is WRAP.

See Also:

EXTRACT
QUERY

SET PFKEY

Syntax:

```
>>--+-------+- PFkey -- n --+----------+--+----------+----------------------><
    |       |               |          |  |          |
    +- SET -+               +- BEFORE -+  +- string -+
```

Description:

Defines the action of a Program Function (PF) Key for the CBLe or SDE window view.

No separating blanks need be specified between the PFKEY option keyword and the pfkey number *n*. e.g. `SET PF13 HELP`

Note: this does not affect the PF Key settings for Class, Default, TitleBars or Borders as viewed and updated via the CBLi KEYS command.

SET PFKEY takes effect at the View level.

Parameters:

n
: Program Function Key number (1 to 24).

`BEFORE`
: The action assigned to the Program Function Key will be executed **before** the contents of the command line and also before any outstanding changes to the file are committed to the in-storage copy of the edited data.
Default is to execute the PFKey action **after** interpreting all other actions (command line contents, text changes, etc.)

`string`
: The command string to be assigned to the PF Key. If *string* is not specified, the PF Key becomes unassigned at the Window level.
Note: the PF Key may still perform actions within the window view if it is assigned at the Class or Default level.

Example:

`pfkey 3 undo`
: Assign UNDO command to PF03.

`pf8 macro hs`
: Assign MACRO HS to PF08.

See Also:

EXTRACT
QUERY

SET POINT

Syntax:

```
>>--+-------+- Point -- .name --+-------+--><
    |       |                   |       |
    +- SET -+                   +- OFF -+
```

Description:

Assign or unassign a name to the focus line for subsequent use as a line target.

A line may be assigned more than one name, however, the same name may not be assigned to more than one line in the current file. Where the name is already assigned to a line in the currunt file, it is unassigned from that line and reassigned to the focus line.

A line name remains assigned to a line following changes to the text or a MOVE command.

SET POINT takes effect at the File level.

Parameters:

`.name`	The name to be assigned to or unassigned from the focus line. The preceding "." (dot) is mandatory.
`OFF`	Unassign the specified name from the focus line.

Example:

`point .joe`	Assign the name ".joe" to the focus line.
`p .sale off`	Unassign the name ".sale" from the focus line.

See Also:

EXTRACT
QUERY

SET PREFIX

Syntax:

```
>>--+-------+- PREfix --+- ON --+--+---------+--+----------+--><
    |       |           |       |  |         |  |          |
    +- SET -+           +- OFF -+  +- Right -+  +- n_cols -+
                                   |         |
                                   +- Left --+
```

Description:

PREFIX defines whether the prefix area is displayed and whether it is displayed on the left of the window view or on the right.

The prefix area displays the line number of lines in the window view.

Prefix area commands are entered in the prefix area.

SET PREFIX takes effect at the View level.

Parameters:

`ON` `OFF`	ON displays the prefix area, OFF suppresses display of the prefix area. Initially, PREFIX is ON.
`RIGHT` `LEFT`	Display the prefix area at the right or left edge of the window view. If neither RIGHT nor LEFT are specified, the position of the prefix area last defined in the current CBLe view, is unchanged. Initially, the prefix area is on the RIGHT.
n_cols	Width of the prefix area in number of columns. Default is 6.

Example:

`pre on left`	Display the prefix area on the left of the current window view.

See Also:

EXTRACT
QUERY
Prefix Commands

SET PSCOPE

Syntax:

```
>>--+-------+- PSCOPE --+- All -----+-->< 
    |       |           |           |
    +- SET -+           +- Display -+
```

Description:

PSCOPE defines whether lines with selection levels that exclude them from the current display, are respected or ignored by CBLe prefix area and ISPF edit line commands.

SET PSCOPE ALL is default for interface ISPF. SET PSCOPE DISPLAY is default for interface CBLe.

SET PSCOPE takes effect at the View level.

Parameters:

`ALL`	Respect excluded lines for all CBLe prefix area and ISPF line commands. The only exception to this is '/' (slash - x'61') which makes the line the current (top) line of the display.
`DISPLAY`	Execute CBLe prefix area and ISPF line commands against displayed lines only. i.e. Ignore excluded lines.

See Also:

EXTRACT
QUERY
SET SCOPE

SET RANGE

Syntax:

```
>>--+-------+-- RANGe --+-- line-target1 -- line-target2 --+-->< 
    |       |           |                                  |
    +- SET -+           +----------- BLock ----------------+
```

Description:

Restricts the edit view to a range of consecutive lines within which CBLe commands are to operate.

The top and bottom lines of the range are equal to the lines referenced by line-target1 and line-target2 respectively, or the top and bottom lines of a marked block.

SET RANGE does not change line numbering but those lines whose line numbers fall outside the range, are removed from the edit view and are not affected by subsequent edit commands. However, CBLe commands that write data to disk (e.g. SAVE, FILE) will write all the file data regardless of the range of lines in view.

Where the top line of the range is not the first line of the file, the "Top of File" line becomes "Top of Range (Line=nnn)". Similarly, where the bottom line of the range is not the last line of the file, the "End of File" line becomes "End of Range (Line=mmm)".

SET RANGE takes effect at the View level.

Parameters:

`line-target1`
`line-target2`
Line targets identifying the first and last lines of the range.

Relative line number targets, string targets and line class targets are always determined relative to the current focus line. Unlike the LOCATE command, the focus line is not changed to be the line found by line-target1. Therefore line-target2 will be determined from the same focus line as target-line1. This may result in a non-ascending range of lines which would return the following error message:

`EDT064E Range is invalid: end `*`nn`*` is less than top `*`mm`*`.`

`BLOCK`
Use the top and bottom lines of a marked block as the limits of the range.

Examples:

```
set range :10 :20
range 10 /==SELCEnd==/
range -* *
ran -/Section 1/ /Section 2/
ran :22 word /idle/
ran block
```

See Also:

EXTRACT
QUERY
SET ZONE

SET RECFM

Syntax:

```
>>--+-------+- RECFM --+- V -+---------------------------------------------><
    |       |          |     |
    +- SET -+          +- F -+
```

Description:

Change the record format of the currently edited in-storage data.

The **Fmt=** value in the status bar and RECFM value in the title bar of the CBLe or SDE edit window view are updated to reflect the change.

Following a save operation, the edited data will be saved to disk using the changed record format value. For MVS, RECFM may only be changed where the fileid of the currently edited data is not that of an existing data set. (i.e. a new data set will be allocated when the data is saved.)

Note: SET RECFM is not yet implemented for VSE.

SET RECFM takes effect at the File level.

Parameters:

`V`
: Record format is variable.

`F`
: Record format is fixed.

See Also:

EXTRACT
QUERY

SET RESERVED

Syntax:

```
>>--+-------+- RESERved -- line --+- OFF ---------------+--------------------><
    |       |                     |                     |
    +- SET -+                     +- colour -+----------+
                                             |          |
                                             +- string -+
```

Description:

Reserve a line and optionally insert a coloured text string in the CBLe or SDE window view for the current file. The line number is specified as being relative to the top, middle or bottom of the document window.

A reserved line may be implemented to display useful information such as the structure of text fields in the current file.

SET RESERVED takes effect at the File level.

Parameters:

line
: Line number of the reserved line relative to the top, middle or bottom of the document window.

  Line numbers relative to the top of the document window are specified as positive integers. e.g. 2, 8

  Line numbers relative to the middle of the document window are specified as offsets from M. e.g. M+1, M-3

  Line numbers relative to the bottom of the document window are specified as negative integers. e.g. -6, -10

`OFF`
: Switch off RESERVE for the specified line number.

colour
: The colour assigned to the reserved line. Supported colours and extended highlighting are described in SET COLOUR.

string
: Text string to be inserted in the reserved line. Where no string is specified, the reserved line will be blank.

Example:

`reserve  m-2   yellow  rev  |-Key-| |----Name----| |---------email---------|`
: Define a reserved line 2 lines up from the middle of the document window. The reserved line will contain a text string describing columns of data coloured in reverse video yellow.

`reserve 3  blue  F1=Top  F2=Bottom  F3=Quit`
: Define a reserved line 3 lines down from the top of the document window. The reserved line will contain a text string describing PF Keys coloured in blue.

See Also:

EXTRACT
QUERY
SET COLOUR

SET SCALE

Syntax:

```
>>--+-------+- SCALe --+- ON --+--+--------+--><
    |       |          |       |  |        |
    +- SET -+          +- OFF -+  +- line -+
```

Description:

SCALE defines whether the scale line is displayed in the CBLe document window and optionally the line number at which the scale line is displayed.

The line number is specified as being relative to the top, middle or bottom of the document window.

The scale line not only identifies the column numbers, but displays the current left and right ZONE columns as "<" (less than) and ">" (greater than) respectively, and the current column as "|" (vertical bar).

SET SCALE takes effect at the View level.

Parameters:

`ON`

`OFF`
: ON displays the scale line, OFF suppresses display of the scale line. Initially, SCALE is ON.

`line`
: Line number of the scale line relative to the top, middle or bottom of the document window.

  Line numbers relative to the top of the document window are specified as positive integers. e.g. 2, 8

  Line numbers relative to the middle of the document window are specified as offsets from M. e.g. M+1, M-3

  Line numbers relative to the bottom of the document window are specified as negative integers. e.g. -6, -10

Example:

`set scal on 2`	Display the scale line 2 lines down from the top of the document window.
`scale off`	Suppress display of the scale line.

See Also:

EXTRACT
QUERY

SET SCOLOR, SET SCOLOUR

Syntax:

```
                                              +- Green -----+ +- NONe -----+
                                              |             | |            |
>>--+-------+--+-- SCOLour -+-+- line-target -+-------------+-+------------+-+-----------+-+--><
    |       |  |            | |               |             | |            | |           | |
    +- SET -+  +-- SCOLor --+ |               +- Blue ------+ +- BLInk ----+ +--- name --+ |
                              |               |             | |            |               |
                              |               +- Default ---+ +- REVvideo -+               |
                              |               |             | |            |               |
                              |               +- Pink ------+ +- Uscore ---+               |
                              |               |             |                              |
                              |               +- Red -------+                              |
                              |               |             |                              |
                              |               +- Turquoise -+                              |
                              |               |             |                              |
                              |               +- White -----+                              |
                              |               |             |                              |
                              |               +- Yellow ----+                              |
                              |                                                            |
                              +---------------------+-- name --+-- OFF --------------------+
                                                    |          |
                                                    +--- * ----+
```

Description:

Assign and unassign colours to all occurrences of the specified line-target string. The SET SCOLOUR command is influenced by the current setting for CASE and ZONE.

All SET SCOLOUR commands that have been issued for the file, are stored and applied in order to any new text entered in the file.

SET SCOLOUR takes effect at the File level.

Parameters:

`line-target`
: The line-target string to be coloured.

`BLUE`
`DEFAULT`
`GREEN`
`PINK`
`RED`
`TURQUOISE`
`WHITE`
`YELLOW`
: Supported colours. If DEFAULT is specified the 3270 hardware default is used.

`BLINK`
`NONE`
`REVVIDEO`
`USCORE`
: Extended highlighting. The colour may blink, be displayed in reverse video or be underlined. Default is NONE.

`name`
: Unique name by which the SET SCOLOUR entry is referenced.

Name may be enclosed within delimiter characters which are included in QUERY and EXTRACT SCOLOUR output but do not form part of the name itself.

Name may be used to reference a specific SET SCOLOUR entry to set it off.

If the name used already exists for a prior SET SCOLOUR command, then the original SET SCOLOUR entry is set off and replaced by the new definition.

If name is not specified, it defaults to being the line-target string.

Note: name does not include any special keywords used in the line-target (i.e. WORD, PREFIX or SUFFIX.)

`*`
: (Asterisk) references all SET SCOLOUR entries.

`OFF`
: Remove a SET SCOLOUR entry and undo any colouring caused by that entry.

Examples:

```
set scolour /*/ green none asterisk
scol help yel uscore "help items"
scol pre 'sup' pink blink
```

See Also:

EXTRACT
QUERY
SET CASE
SET COLOUR
SET LCOLOUR
SET ZONE

SET SCOPE

Syntax:

```
>>--+-------+- SCOPE --+- All -----+--><
    |       |          |           |
    +- SET -+          +- Display -+
```

Description:

SCOPE defines whether lines with selection levels that excludes them from the current display, are respected or ignored by most CBLe commands.

On successful execution of an ALL line-target command, SCOPE DISPLAY is set automatically.

SET SCOPE takes effect at the View level.

Parameters:

`ALL`	Respect excluded lines for all CBLe commands.
`DISPLAY`	Execute CBLe commands against displayed line only. i.e. Ignore excluded lines. Exceptions to this are SAVE, FILE and SORT which operate on all lines in the display, regardless of SCOPE setting. Initially, SCOPE is DISPLAY

See Also:

EXTRACT
QUERY
ALL
SET DISPLAY
SET SELECT
SET SHADOW

SET SELECT

Syntax:

```
                                    +- 1 ------------+
                                    |                |
>>--+-------+- SELect --+-  n -+--+----------------+-->&lt;
    |       |            |     |  |                |
    +- SET -+            +- +n -+  +- group-target -+
                         |     |
                         +- -n -+
```

Description:

SELECT changes the selection level of all lines in the group-target area.

Every line of an edited file window is initially assigned a selection level of 0. The selection level of a line may be changed to any number in the range 0 to 255.

SET DISPLAY and SET SCOPE DISPLAY rely on the selection level of lines to determine lines displayed and lines on which CBLe commands will operate.

The selection level of a line may also be changed automatically by the ALL command and when using the X or S prefix area command.

SET SELECT takes effect at the File level.

Parameters:

`n`	The selection level number set for all lines in the target area.
`+n` `-n`	The "+" (plus) and "-" (minus) prefix indicates that the current selection levels of lines in the target area are to be incremented or decremented by the value n. If either of these result in a selection level less than 0 or greater than 255, then selection level set will be 0 or 255 respectively.
`group-target`	Group target condition defining the target area.

Example:

`sel 8 /SELC/`	Set selection level 8 to the focus line and lines up to, but not including, the first line to contain the string "SELC".
`select +1`	Add 1 to the selection level of the focus line.

See Also:

EXTRACT
QUERY
ALL
SET DISPLAY
SET SCOPE

SET SHADOW

Syntax:

```
>>--+-------+- SHADow --+- ON --+-->< 
    |       |           |       |
    +- SET -+           +- OFF -+
```

Description:

SHADOW defines whether a shadow line is displayed to represent a line or group of lines that are excluded from the current CBLe view.

Each shadow line indicates the number of consecutive lines that have been excluded from that position in the CBLe view.

SET SHADOW takes effect at the View level.

Parameters:

ON OFF	Set SHADOW ON or OFF. Default is ON.

See Also:

EXTRACT
QUERY
ALL
SET SELECT
SET DISPLAY

SET SIZEWARNING

Syntax:

```
>>--+-------+-- SIZEWarning --+---------+--+----------+--><
    |       |                 |         |  |          |
    +- SET -+                 +-- OFF --+  +-- nnn  --+
                                           |          |
                                           +-- nnnK --+
                                           |          |
                                           +-- nnnM --+
```

Description:

When a file is opened for edit, the contents of the file are first loaded into storage.

SIZEWARNING defines the maximum size of any file that may be edited without opening a warning popup window that prompts the user to continue the file edit.

SIZEWARNING is only effective where the file size in bytes can be determined by CBLe prior to load. i.e. PDS(E) member sizes cannot be checked.

The purpose of the file size warning threshold is to stop a user from accidentally editing a very large file.

CBLe SET SIZEWARNING overrides the Edit.SizeWarning value set by the SYSTEM or USER CBLIINI file. If a SizeWarning value has not been set in either CBLiINI file, the default value is 1M (one megabyte).

SET SIZEWARNING takes effect at the Global level.

Parameters:

ON OFF	Enable or Disable file size checking prior to load. Default is ON.
nnn nnnK nnnM	Number of bytes defining the maximum size of any file that can be loaded before the file size warning popup window is opened. This value may be specified as a number of bytes (nnn), number of kilobytes (nnnK) or a number of megabytes (nnnM).

Example:

```
sizew  on  10K
```

When a new file is edited, open a warning popup window if its size exceeds 10 kilobytes.

See Also:

EXTRACT
QUERY
SET LOADWARNING

SET STAY

Syntax:

```
>>--+-------+- STAY --+- ON --+--><
    |       |         |       |
    +- SET -+         +- OFF -+
```

Description:

STAY defines whether the focus line is changed for the following:

1. An unsuccessful LOCATE or CLOCATE command with WRAP OFF. With STAY ON the focus line is unchanged. With STAY OFF the End of File line (or Top of File line for backward searches) becomes the focus line.
2. A successful CHANGE, SET SELECT or SORT command. With STAY ON the focus line is unchanged. With STAY OFF, the last line scanned or affected by the command becomes the new focus line.

SET STAY takes effect at the View level.

Parameters:

ON OFF	Set STAY ON or OFF. Default is ON.

See Also:

EXTRACT
QUERY

SET STREAM

Syntax:

```
>>--+-------+- STReam --+- ON --+--><
    |       |           |       |
    +- SET -+           +- OFF -+
```

Description:

STREAM defines whether a search for a column target streams over multiple lines or is restricted to the focus line only.

Note: SET STREAM only affects CLOCATE and CDELETE commands as these are the only commands to use column-target.

SET STREAM takes effect at the View level.

Parameters:

ON OFF	Set STREAM ON or OFF. Default is ON.

See Also:

EXTRACT
QUERY

SET SYNONYM

Syntax:

```
>>--+-------+-- SYNonym --+-- name ---+-------+-- action --+-------><
    |       |             |           |       |            |
    +- SET -+             |           +-- n --+            |
                          |                                |
                          +----------+-- ON ---+-----------+
                                     |         |
                                     +-- OFF --+
```

Description:

SET SYNONYM has the following functions:

1. Controls CBLe synonym processing.

   With SYNONYM ON, CBLe checks any command verb to be executed as being a defined synonym name. The exception to this is when the command issued is done so via a CBLe macro. In this case, the SYNEX command must prefix the command to be executed in order to force CBLe's synonym processing.

   With SYNONYM OFF, no synonym checking occurs.

   CBLe's synonym checking may be bypassed by prefixing the command with the COMMAND command.
2. Defines new synonym names and associated actions.

   The name token is assigned the specified action. The name may be defined with a minimal truncation of n characters. Therafter, only the first n characters of the name need be entered to execute the associated action.

   Each name/action definition is stored until the MDI application is closed.

SET SYNONYM ON|OFF takes effect at the View level.
SET SYNONYM name takes effect at the Global level.

Parameters:

`ON` `OFF`	Set SYNONYM processing ON or OFF. Default is ON.
`name`	Synonym name which may be equal to an existing CBLe command or macro name.
`n`	Number of characters that CBLe will accept as the minimal truncation for name.
`action`	CBLe command stream.

Examples:

```
set synonym off
synonym find 1 locate
syn delblank imm 'ext/flscr/';'nomsg all blank';'del *';'all';':'flscreen.1
```

See Also:

EXTRACT
QUERY

SET THIGHLIGHT

Syntax:

```
                                         +-- ALL ---+
                                         |          |
>>--+-------+-- THIGHlight --+-- ON ---+--+----------+-----><
    |       |                |         |  |          |
    +- SET -+                +-- OFF --+  +-- FIRST -+
```

Description:

Controls whether the target of a LOCATE or CLOCATE command is highlighted.

The target remains highlighted until another LOCATE or CLOCATE command is executed or until text within the file is changed in any way.

The colour used to highlight the target is determined by SET COLOUR THIGHLIGHT.

SET THIGHLIGHT takes effect at the View level.

Parameters:

`ON` `OFF`	Set THIGHLIGHT ON or OFF. Default is ON.
`FIRST` `ALL`	Highlight all occurrences of the target string or just the first. Default is ALL.

Examples:

```
thighlight on first

thighlight off
```

See Also:

EXTRACT
QUERY
CLOCATE
LOCATE
SET COLOUR

SET UNDOING

Syntax:

```
>>--+-------+- UNDOING --+- ON --+--+------------+--><
    |       |            |       |  |            |
    +- SET -+            +- OFF -+  +- n --+-----+
                                           |     |
                                           +- k -+
```

Description:

UNDOING defines whether the UNDO (and REDO) facility is enabled, the number of change levels that CBLe will attempt to maintain and the maximum amount of storage CBLe can allocate in order to store this information.

The third number following "Alt=" on the status line displays the current number of stored change levels.

The change level count is incremented by any undoable command. An undoable command is any command that changes the data in the document area or the flag bits of a line. Flag bits include a line's selection level and its tagged, changed and new indicators.

Note: typing text on a line in the document window is effectively treated as a CINSERT or CREPLACE command. Therefore, changing or adding text to multiple lines is equivalent to multiple CINSERT or CREPLACE commands each resulting in a different change level.

Multiple changes made to a file as a result of a macro execution are considered to be one change level only.

CBLe is informed of any changes to the 3270 terminal when an Attention ID (AID) is generated (e.g. on hitting the Enter key or any of the PF Keys). It is only then that changes to the file are committed to CBLe memory and the change level is updated. Therefore, where changes have been made to text on multiple lines, CBLe has no indication as to the order in which the lines were changed and so assigns a change level to each updated line in ascending order of line number.

SET UNDOING takes effect at the File level.

Parameters:

`ON` `OFF`	Set UNDOING ON or OFF. Default is ON.
`n`	Number of change levels maintained by CBLe. If this value is exceeded, CBLe drops the oldest undoable change level.

`k` Maximum amount of storage (KB) that may be obtained by CBLe for storing undo information.

See Also:

EXTRACT
QUERY

SET VARBLANK

Syntax:

```
>>--+-------+- VARblank --+- ON --+--><
    |       |             |       |
    +- SET -+             +- OFF -+
```

Description:

VARBLANK defines whether a single blank, specified as part of a line-target or column-target search string, represents one or more blanks.

SET VARBLANK takes effect at the View level.

Parameters:

ON OFF	Set VARBLANK ON or OFF. Default is OFF.

Example:

`varblank on` A single blank in a line or column target search string represents one or more blanks. CLOCATE /ab c/ will successfully match the following:

```
ab c
ab  c
ab        c
```

See Also:

EXTRACT
QUERY

SET VIEW

Syntax:

```
>>--+-------+- VIEW --+- CHAR -+--><
    |       |         |        |
    +- SET -+         +- HEX --+
```

Description:

VIEW the contents of the current CBLe window in character or hexadecimal.

For CHAR display, each line displayed occupies 1 line, the line text in character format.

For HEX display, each line displayed occupies 4 lines as follows:

Line 1	The line text in character format.
Line 2	The zone digit of each character. (Hex line 1)
Line 3	The numeric digit of each character. (Hex line 2)
Line 4	A separator line of "-" (minus) signs.

Both the character and hex lines may be edited normally. Changes made to the character line are reflected in the hex lines when the changes are committed. Similarly for changes made to the hex lines.

SET VIEW takes effect at the View level.

Parameters:

```
CHAR
HEX
```
Set VIEW HEX or CHAR. (Default is CHAR.)

See Also:

EXTRACT
QUERY

SET WINNAME

Syntax:

```
>>--+-------+-- WINNAMe --+- FRAme -+-- name ------------------------------><
    |       |             |         |
    +- SET -+             +- VIew --+
```

Description:

Every window in the CBLi environment is allocated a unique window name which is displayed in the window title bar if "Show window names" is in effect (CBLi command WINDOWNAMES.) Similarly, the Window List (CBLi command WINDOWLIST) also displays each window's window name.

SET WINNAME is primarily used in CBLe/SDE macros, allowing the user to assign a new window name to the current CBLe or SDE edit window view or frame (MDI parent window.) The new name will replace the existing window name.

The window name may be used in certain CBLi CLI window commands (e.g. MAXIMISE, MINIMISE, KEYS, COMMANDLINE, etc.) to reference the window to which the command should apply.

Parameters:

```
FRAME
VIEW
```
Update the frame or edit view window name.

name

Name to assign to the specified window. This may be of length up to 256 characters.

Examples:

```
winname  view  PROJECT_VIEW1
```
Update the edit view window name.
```
winname  fra   PROJECT_FILES
```
Update the CBLe frame window name.

See Also:

EXTRACT
QUERY
WINDOW
SET WINPOS
SET WINSIZE

SET WINPOS

Syntax:

```
                           +- VIew --+
                           |         |
>>--+-------+-- WINPOS --+---------+-- row -- col --><
    |       |            |         |
    +- SET -+            +- FRAme -+
```

Description:

Position the current edit view at the specified row and column within the frame (MDI parent) client area, or position the current frame (MDI parent) window at the specified row and column of the screen.

A window's positional coordinates refer to the row x column position of the window's system menu button (found on the extreme left of the title bar.) It is this coordinate that is positioned at the new row x column position.

The lowest position within an MDI client area is row 1, column 1, which corresponds to the top left position below the window's menu bar.

The lowest position within the screen is row 1, column 2, which corresponds to the top left position occupied by the CBLi main window's system menu button.

A window cannot be positioned entirely outside its parent (MDI or CBLi main) window. If large row x column values are used, CBLe will ensure that at least 5 characters of the window's title bar remains in view.

Parameters:

`FRAME` `VIEW`	Position the frame or edit view window. Default is VIEW.
`row`	A positive, non-zero number specifying the row number at which the window is to be positioned.
`col`	A positive, non-zero number specifying the column number at which the window is to be positioned.

Examples:

`winpos view 5 10`
Position the edit view at row 5, column 10 of the current frame's client area.

`winpos 10 15`
Position the edit view at row 10, column 15 of the current frame's client area.

`winpos frame 1 2`
Position the frame window at row 1, column 2 of the screen.

See Also:

EXTRACT
QUERY
WINDOW
SET WINNAME
SET WINSIZE

SET WINSIZE

Syntax:

```
                            +- VIew --+
                            |         |
>>--+-------+-- WINSIZe --+---------+-- rows -- cols --><
    |       |             |         |
    +- SET -+             +- FRAme -+
```

Description:

Resize the current edit view or frame (MDI parent) window to the specified number of rows and columns.

CBLe does not allow an edit view display to be resized to a window less than 5 rows deep by 10 columns wide or to a window size greater than that of the MDI parent window.

Similarly, CBLi does not allow an MDI parent window display to be resized to a window of less than 10 rows deep by 10 columns wide or to a window size greater than that of the CBLi main window.

If WINPOS rows/cols values are used that exceed these limits, the window is resized to the allowable limit.

Parameters:

FRAME VIEW	Resize the frame or edit view window.
rows	A positive, non-zero number specifying the number of rows to which the window is to be resized.
cols	A positive, non-zero number specifying the number of columns to which the window is to be resized.

Examples:

```
winsiz view 28 80
```

```
winsize fra 40 100
```

See Also:

EXTRACT
QUERY
WINDOW
SET WINNAME
SET WINPOS

SET WRAP

Syntax:

```
>>--+-------+- WRAP --+- ON --+--><
    |       |         |       |
    +- SET -+         +- OFF -+
```

Description:

WRAP defines whether a scan for a string line-target continues from the Top of File after End of File has been reached. Similarly for backward searches where the scan may continue from the End of File when Top of File has been reached.

WRAP affects the LOCATE command and the CLOCATE command when STREAM is ON.

Where WRAP is OFF and the End of File or Top of File is reached, then the following message is returned:

```
EDT005E Target not Found
```

Where WRAP is ON and the target is found after a wrap has occurred, then the following message is returned:

```
EDT018I Wrapped ...
```

SET WRAP takes effect at the View level.

Parameters:

ON OFF	Set WRAP ON or OFF. Default is OFF.

See Also:

EXTRACT
QUERY

SET ZONE

Syntax:

```
>>--+-------+- Zone -- n1 -- n2 --><
    |       |
    +- SET -+
```

Description:

ZONE defines the leftmost and rightmost column between which all string searches operate. i.e. line-target, column-target, group-target and CHANGE strings.

The left and right zone columns are represented in the scale line as "<" (less than) and ">" (greater than) respectively.

SET ZONE takes effect at the View level.

Parameters:

`n1` Left zone column. Initially, n1 is 1.

`n2` Right zone column. "*" (asterisk) may be specified to indicate the truncation column which, by default, is equal to the LRECL. Initially, n2 is "*" (asterisk).

Example:

`z 20 26`	Set the left zone column to column 20, the right zone column to column 26.
`z 10 *`	Set the left zone column to column 10, the right zone column to the truncation column.

See Also:

EXTRACT
QUERY

Prefix Commands

The following commands can be entered in the prefix area of an edit window:

.xxx	Set a line pointer (line name). **Note:** unlike ISPF EDIT these line names are not displayed in the prefix area. One reason for this is that a line can have multiple names.
/	Make this line the current line.
<(n) <<(n)	Shift a line or block of lines n columns to the left. "(" (left parenthesis) and "((" may be used as an alternative.
>(n) >>(n)	Shift a line or block of lines n columns to the right. ")" (right parenthesis) and "))" may be used as an alternative.
A(n)	Add a blank line or a block of n blank lines.
C(n) CC	Mark a line or a block of lines for copying.
D(n) DD	Mark a line or a block of lines for deleting.
F	Make this line the target for a move or copy (move or copy lines Following this line).
HEX	Opens a hex dump view of the line.
LC(n) LCC	Mark a line or a block of lines for lower casing.
M(n) MM	Mark a line or a block of lines for moving.
MB	Mark a corner of a box block.
ML	Mark a limit of a line block.
P	Make this line the target for a move or copy (move or copy lines Previous to this line).
R(n) RR(n)	Replicate (duplicate) a line or a block of lines n times.
S(n)	Show a number of excluded lines.
T	Tag a single line.
UC(n) UCC	Mark a line or a block of lines for upper casing.
X(n) XX	Mark a line or a block of lines for exclusion from the display.

Function Keys

3270 Program Function Keys (PFKeys) may be assigned to CBLi and CBLe commands.

The default CBLe function keys may be tailored in the **(Edit)** and **(ISPFEdit)** sections of the SYSTEM and USER CBLiINI files.

Following startup of CBLi, individual CBLe PFKeys may be assigned new definitions using the CBLi KEYS command or the CBLe SET PFKEY command.

Note that SET PFKEY only affects PFKeys at the CBLe edit view level (Window) but may be used in the PROFILE macro to selectively tailor PFKeys based on specific criteria. However, KEYS may be used to tailor PFKey definitions at any of the supported levels (Window, Class, Default, Caption and Border.)

The CBLi **KEYS** command may also be used to open the Function Keys window to display, and optionally update, the current function key settings. Alternatively, the CBLe commands, QUERY PFnn and EXTRACT PFKEY, may be used to obtain individual PFKey settings.

The CBLe text editor maintains two sets of PFKey definitions, one for each of the CBLe and ISPF edit interfaces.

The CBLe editor program default function keys for INTERFACE=CBLE are:

PF1	sos lineadd	Add a line following the cursor line.
PF2	duplicate	Duplicate the cursor line.
PF3	quit	Quit the file (you will be prompted to save any changes).
PF4	cmdtext	Execute CBLi SELECT command on the focus line.
PF5	macro block up major	Scroll up the file so that the first line previous to the current line that contains '***' or '===', becomes the current line.
PF6	macro block down major	Scroll down the file so that the first line following the current line that contains '***' or '===', becomes the current line.
PF7	backward	Scroll the window display backwards 1 page towards the top of the file.
PF8	forward	Scroll the window display forwards 1 page towards the bottom of the file.
PF9	MDINext	Place focus on the next MDI child window. (Edit view or list.)
PF10	left half	Scroll the window display half the width of the edit view to the left.
PF11	right half	Scroll the window display half the width of the edit view to the right.
PF12	retrieve -	Retrieve the last command to the command line.
PF13	sos linedel	Delete the line containing the cursor.
PF14	spltjoin	If the cursor is followed by non-blank data on the line then split the line at the cursor. Otherwise join the cursor line with the line following.
PF15	mark box	Mark the limit of a box block at the cursor.
PF16	mark line	Mark the limit of a line block.
PF17	ECommand copy block	Copy the current marked block to the cursor.
PF18	ECommand move block	Move the current marked block to the cursor.
PF19	ECommand delete block	Delete the current marked block.
PF20	overlaybox	Overlay the current cursor position with the contents of the currently marked block.
PF21	PrevMainWindow	Place focus on the previous CBLi main window. (Focus is removed from CBLe.)
PF22	undo	Undo the most recent change.
PF23	redo	Redo the most recently undone change.
PF24	ECommand reset block	Unmark the block.

The CBLe editor program default function keys for INTERFACE=ISPF are:

PF1	sos lineadd	Add a line following the cursor line.
PF2	duplicate	Duplicate the cursor line.
PF3	end	Quit the file. (Prompt to SAVE depends on the current setting of AUTOSAVE.)
PF4	cmdtext	Execute CBLi SELECT command on the focus line.
PF5	rfind	Locate search string defined by last FIND or CHANGE command.

PF6	rchange	Repeat the change requested by the last CHANGE command.
PF7	up	Scroll the window display upwards by an amount determined by the scroll field or specified on the command line.
PF8	down	Scroll the window display downwards by an amount determined by the scroll field or specified on the command line.
PF9	MDINext	Place focus on the next MDI child window. (Edit view or list.)
PF10	left	Scroll the window display to the left by an amount determined by the scroll field or specified on the command line.
PF11	right	Scroll the window display to the right by an amount determined by the scroll field or specified on the command line.
PF12	retrieve -	Retrieve the last command to the command line.
PF13	sos linedel	Delete the line containing the cursor.
PF14	spltjoin	If the cursor is followed by non-blank data on the line then split the line at the cursor. Otherwise join the cursor line with the line following.
PF15	mark box	Mark the limit of a box block at the cursor.
PF16	mark line	Mark the limit of a line block.
PF17	ECommand copy block	Copy the current marked block to the cursor.
PF18	ECommand move block	Move the current marked block to the cursor.
PF19	ECommand delete block	Delete the current marked block.
PF20	overlaybox	Overlay the current cursor position with the contents of the currently marked block.
PF21	PrevMainWindow	Place focus on the previous CBLi main window. (Focus is removed from CBLe.)
PF22	undo	Undo the most recent change.
PF23	redo	Redo the most recently undone change.
PF24	ECommand reset block	Unmark the block.

Note that, if the following conditions are true, the contents of the command line is concatenated to the definition of the function key and the result executed as a single command.

- The function key command is not set to execute before processing any user screen inputs.
  This is determined by the status of the **BEFORE** field for the PFKey definition in the Function Keys window.
- INTERFACE=ISPF is set for the current CBLe edit view.
- The concatenation of the PFKey function and the command line contents does not result in an invalid ISPF scroll command (UP, DOWN, LEFT, RIGHT.)
  Where an invalid ISPF scroll command is the result, then if the first parameter on the scroll command begins with a numeric character, then an error is returned. Otherwise the contents of the command line are executed before the PFKey function.

Glossary

The following is a glossary of terms used in this document.

3270 Emulator
Third party software that emulates Mainframe 3270 hardware terminals on PC and UNIX based platforms.

CLI
A Command Line Interface is a text based method by which users can execute functions supported by the application.

CBLe
A powerful text editor that runs as an MDI application under CBLi. CBLe supports its own CLI and has been developed based on specifications found in Mansfield Software's KEDIT for Windows.

CBLi
The Interactive environment developed by CBL and supplied as part of SELCOPY and CBLVCAT licensable software products.

CBLiINI
File containing configuration options for CBLi. The SYSTEM CBLiINI file is processed on startup of CBLi and contains options that apply to all users. The USER CBLiINI file contains options specific to each user that may, where appropriate, override options set in the SYSTEM CBLiINI file.

CBLiVTAM
Name of the multi-user version of the CBLi application that executes under VTAM.

CBLVCAT
CBL licensable product that supports VSAM file tuning and VTOC, ICF/VSAM catalog and VSE LABEL reporting. Executes as a batch facility or interactively as a CBLi application.

Edit View
A CBLe MDI document window that contains a display of edited data. If the same file is displayed in multiple windows, then the user has multiple edit views of the file. Each edit view can have a different current line, ARBCHAR setting, ZONE columns, etc.

List Window
A CBLi window containing rows of associated information. List windows support point-and-shoot column sorting; select, sort and filter CLI commands; and prefix area commands.

MDI
Multiple Document Interface is a Microsoft specification for PC applications that enable the user to work with multiple documents at the same time. Each document is displayed in a separate child window within the client area of the application's main (frame) window. Typical MDI applications on PCs include word-processing and spread sheet applications.

MDI Client Area window
The MDI client area window is the display area within an MDI application's frame window. The MDI client area serves as the backround for MDI child windows.

MDI Child/Document Window
An MDI child or document window is opened in an application's client area window each time a document is opened. Each child window has a sizing border, title bar, window menu, minimise, maximise, restore and close buttons. A child window is clipped so that it is confined to the client window and cannot appear outside it.
When a child window is maximized, its client area completely fills the MDI client area window. In addition, the system automatically hides the child window's title bar, and adds the child window's window menu icon and Restore button to the MDI application's menu bar.

MDI Frame Window
An MDI frame window may be considered the main window of an MDI application. It is the parent window of the MDI client area window in which MDI child windows are opened. It has a sizing border, title bar, window menu, minimise, maximise restore and close buttons.

Ring
The set of all **files** being edited within CBLe. It is not the set of all windows opened. e.g. The contents of one file may be displayed in more than one edit view (window.)

SDB
See SELCOPY Interactive.

SELCOPY Interactive (SDB)
An Intergrated Development Environment for SELCOPY (SELCOPY DeBug) that runs as an MDI application under CBLi.

Storage Display Window
A CBLi window containing hexadecimal and character display of areas of storage.